AN EMPIRICAL COMPARATIVE INVESTIGATION OF THE CP/67 AND TSS/360 TIME-SHARING SYSTEM

William Robert Haines

United States
Naval Postgraduate School

THESIS

An Empirical Comparative Investigation
of the CP/67 and TSS/360 Time-Sharing System

by

William Robert Haines

and

James Harold Porterfield, Jr.

Thesis Advisor: G. H. Syms

June 1971

Approved for public release; distribution unlimited.

T139859

An Empirical Comparative Investigation of the CP/67 and TSS/360 Time-Sharing Systems

by

William Robert Haines
Lieutenant, United States Navy
B. S., United States Naval Academy, 1964

and

James Harold Porterfield, Jr.
Lieutenant, United States Navy
B. S., United States Naval Academy, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

ABSTRACT

A set of terminal test programs called a benchmark has been derived for the purpose of comparing the two time-sharing computer systems, CP/67 and TSS/360, in order to determine which one can best meet the present operational requirements at the Naval Postgraduate School. Some of the problems encountered in attempting to design the benchmark are discussed, along with the problems of trying to make a valid comparison of the two systems. The results obtained from a series of tests conducted over a period of six months are compiled, analyzed and presented in a manner which shows that CP/67 is significantly superior in most respects to TSS. In addition to the comparison of the two systems, information is presented which spotlights many of the problems which degrade CP/67, and recommendations for the alleviation of those problems are made.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In recent years the evaluation of computer systems has lagged behind the development of computer hardware and software technology. Computer centers have been persuaded to obtain bigger, faster and newer computers--"the latest thing"-- rather than evaluate their current operations and attempt to maximize their performances. With increased costs, many centers have found it necessary to optimize the operation of the existing installation, or to make detailed comparisons of competing systems when obtaining new equipment. Anytime anyone is presented with a choice of anything as complex as a computer system, he should have some basis or reason for his choice other than chance or intuition. For these reasons, the problem of obtaining satisfactory evaluation is of primary interest in the field of computer science.

While there are benchmarks and measurement devices for comparing batch oriented systems, little work has been done in evaluating time-sharing systems and optimizing their performance. This is due primarily to the many special problems which arise when attempting to evaluate a time-sharing system, and to the lack of any standards by which to compare performance. The problems of optimization and evaluation of time-sharing systems is much more difficult than for its batch counterpart. In a batch operating system, there are relatively few user programs competing for system resources at any one time. Thus, it is relatively simple to collect a set of benchmark programs that are representative of a "typical" load that could be used for evaluating and comparing the computer systems. On the other hand, in a time-sharing system there are many user programs competing for resources and, at any one time,

there many be many requests for a particular resource. Since there are so many combinations of requests, it is very difficult to construct a set of test programs (a benchmark or a set of scripts) that will represent a "realistic load" for comparing two time-sharing systems. For reasons such as these and the fact that nearly every computer facility is unique in the type of work loads that present itself to that particular time-sharing system, there has been no standard benchmark developed to compare and evaluate time-sharing systems in general.

Because of the different nature of the systems, performance measurements present a more difficult obstacle to time-sharing than to a batch system. The primary performance indicators for batch are turn-around and throughput, while the ultimate test of a time-sharing system is its performance as seen by the individual user. While it is easy to calculate the throughput for a time-sharing system, a figure corresponding to turn-around is required. Response time is one solution, however, it is not as straightforward as the measurement of turn-around time in a batch system. In determining an acceptable response time it is not only necessary to consider the response time and its relationship to various job types, but it is also necessary to consider the users reaction to these response times. For example, a user may find a ten second response to one type of request unacceptable while a five minute response to another request is quite acceptable. Along with the individual user's performance criteria, the performance or efficiency of the system as a whole must be studied before any substantial improvements to the system or comparisons with other system can be made.

The problems presented above are of a general nature and apply to any time-sharing system. This thesis is concerned with the procedures that were used to compare the CP/67 and IBM TSS/360 time-sharing systems

on the IBM 360/67 computer system at the Naval Postgraduate School. The primary emphasis is on the development of a benchmark, the results of the comparison and the observation of system parameters that directly affect the operation of a time-sharing system.

II. SPECIFIC OBJECTIVES

At the commencement of this study, both TSS and CP had been used as the time-sharing system at different times. Although TSS was the system in use at the time, a final determination of which system to use at the Naval Postgraduate School had not been made. This situation provided the impetus for the study, and it was undertaken to determine which system provided the best service.

The primary objective was to judge the relative performance of the two systems, but in order to accomplish this a means of gathering information from the systems had to be designed. An outgrowth of the primary objective was to develop a benchmark that was not only suitable for the comparison of the systems under study, but also would provide general procedures which could be expanded to meet the requirements imposed by any time-sharing system or any facility. The basic requirement for the benchmark was that it accurately relect the normal load and at the same time provide sufficient data to make a meaningful comparison of the systems.

A secondary objective was to determine an ideal job mix for whichever system was chosen as superior and use this mix to arrive at standards which would impose limitations on the types and numbers of jobs which could be run in the system at any one time. In this way the system could be maintained at peak efficiency during periods of heavy use.

The final area of interest was the examination of the systems in detail to gain insight into the problem of optimization of a time-sharing system. This consisted of observing the effects of various

jobs on CPU utilization, paging requirements and channel activity. These factors are instrumental in determining the efficiency and behavior of the system.

III. DESCRIPTION OF SYSTEMS

The comparison of TSS/360 and CP/67 was conducted on the IBM 360/67 computer system configured as shown in Figure 1. CP/67, version three, was used with the following hardware except where noted in the remainder of the thesis:

1	2365	core box (256K)
1	2067	central processing unit
	2311	disk drives
	2741	terminal units
	2301	drum

TSS/360 version eight, with schedule table T47 [Doherty, 1970] was configured as follows:

3	2365	core boxes (768KO
2	2067	central processing units
	2314	direct access storage
	2741	terminal units
	2301	drum

Thus, CP/67 had only one-third the core memory, slower disks, and only one processor (CPU). This comparison was made because running CP with these limited resources allowed the simultaneous running of a batch operation under OS/MVT on the other processor with two.core boxes(512K bytes) and the 2314 disk. Thus, if it could be shown that CP could compete with TSS under these conditions, then it would be much more economical to run this installation as a split system rather than as a dual processor system under TSS.

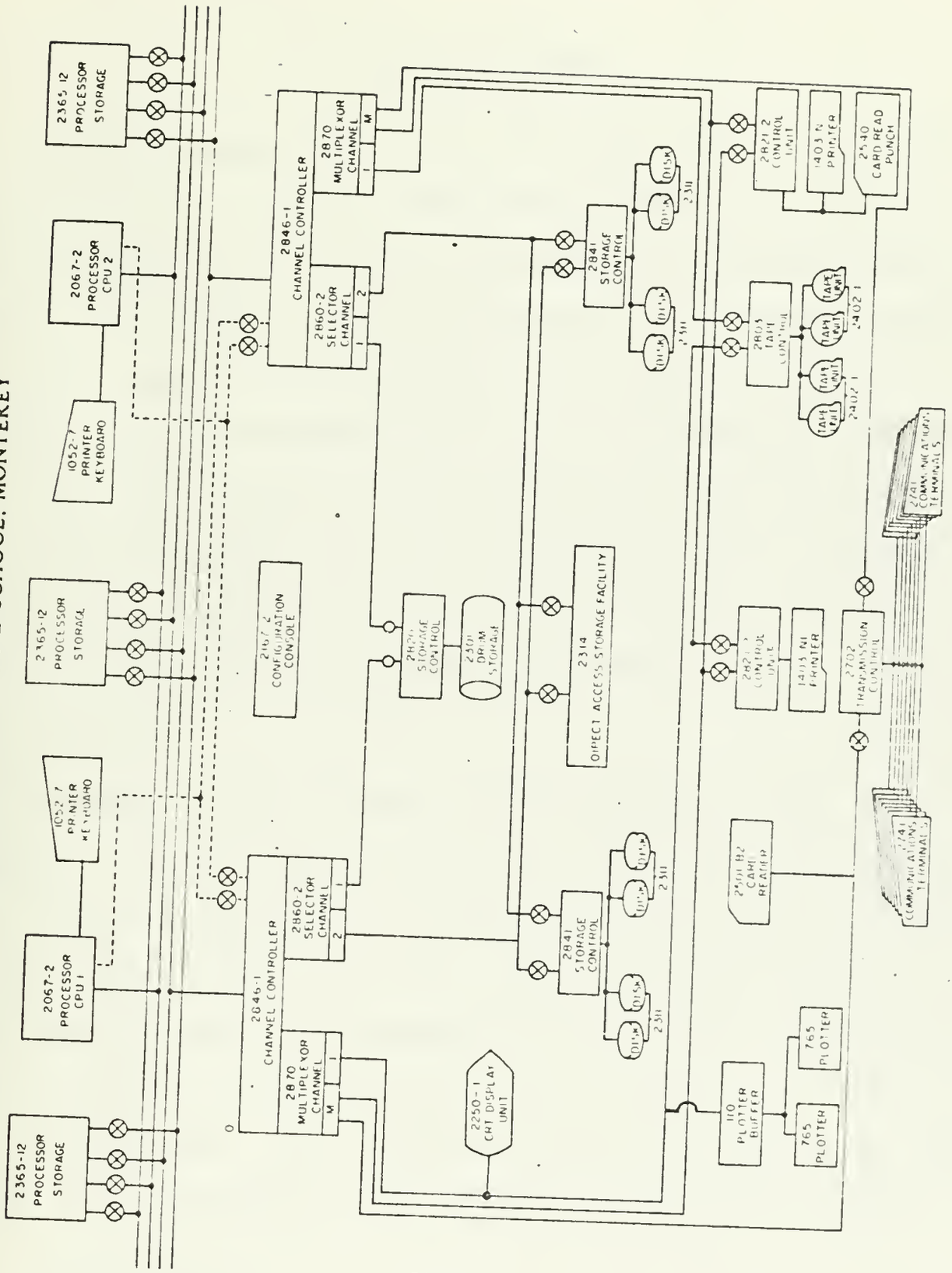
IBM 360. MODEL 67 CONFIGURATION CHART
NAVAL POSTGRADUATE SCHOOL, MONTEREY
2365-12 PROCESSOR STORAGE
2067-2 PROCESSOR CPU 1
2067-2 PROCESSOR CPU 2
1052-7 PRINTER KEYBOARD
2846-1 CHANNEL CONTROLLER
2860-2 SELECTOR CHANNEL
2870 MULTIPLEXOR CHANNEL
2250-1 CRT DISPLAY UNIT
2841 STORAGE CONTROL
2301 DRUM STORAGE
2314 DIRECT ACCESS STORAGE FACILITY
2311 DISK
2821-2 CONTROL UNIT
1403 N1 PRINTER
2540 CARD READ PUNCH
2803 TAPE CONTROL
2402-1
2702 TRANSMISSION CONTROL
2501-B2 CARD READER
110 PLOTTER BUFFER
765 PLOTTER
2067-2 CONFIGURATION CONSOLE

Figure 1.

IV. DESIGN PROCEDURES

In order to compare TSS and CP at this installation it was decided to conduct a series of tests. These tests would be performed under strict conditions so that neither system would be prejudiced one way or another, thus avoiding an unfair determination. Since the two systems were to be compared in order to determine which provided the better service, each test was designed to run on both systems using identical load conditions. In addition each system was run under the same hardware configuration that would normally be used during actual operation at the school. Therefore TSS would run dual processor with three core boxes and CP would run single processor with one core box.

A. JOB MIX

An attempt was made to derive a set of test programs that could represent a "typical" job mix. This turned out to be an insurmountable task, especially since this is an academic environment and the job load changes from one quarter to the next. Additionally with the rapid advancement in the field of programming techniques most situations are not repeated, and therefore not even a cyclic repetition of job load/mix can be expected. Thus, considerable flexibility -probably too much- exists in the selection of a job mix to be used for testing purposes.

B. FACTORS ANALYZED

In comparing the two systems, two main factors were to be scrutinized after each test to determine the best system for use here. First, from the standpoint of computer efficiency, CPU utilization was to be determined. Since low CPU utilization indicates that system is either

not being used to its full capacity or that it is being overloaded to the point of being very wasteful of its resources, maximum CPU utilization would be a dominant factor in determining performance.

The second factor to be considered was the matter of how many users would each system support and still provide acceptable performance. Because this is a time-sharing system the number of users supported is a very critical factor, yet pure numbers are useless unless each user can be accommodated in a manner which will be acceptable to him.

A third factor which needed to be determined was the overload point of the system and, if it was reached, how long it would take each system to recover from such a circumstance.

C. INITIAL BENCHMARK

The initial approach to this somewhat awesome problem was to design a set of preliminary programs and to develop this set into a valid and workable benchmark. The method used in this series of tests was that of the stimulus approach, described by Karush [Karush, 1969] on the development of a benchmark for the ADAPT-50 time-sharing system at Systems Development Corporation. Since reliable statistics were not available in past usage of either system at this installation, the initial selection of a job mix was made rather arbitrarily.

For the initial test, three types of jobs were chosen - compilations, executions and edits - and four basic program types were used:

1. PL/1 compilation - PL/1 is the most commonly used language by Computer Science students and it was felt that the PL/1 compiler would be one of the more important factors in the performance of the system due to the heavy paging demand by the compiler.

2. FORTRAN compilation - FORTRAN is the most commonly used language at the school as a whole.

3. FORTRAN execution - The language is not important here because the interest was to have a computer bound program.

4. Editing - An exec routine was written with CMS commands to simulate an editing user. The program being edited was the same FORTRAN program that was compiled.

The selection of edit-to-run (compile and/or execute) ratio was made in accordance with the type of load desired on the system at any particular time, with lighter loads having very high ratios and heavier loads having proportionally smaller ratios. In attempting to influence the CPU utilization, the compile-to-execute ratio within the non-edit programs was altered with the lower ratios giving more CPU time due to the compute bound nature of the execution program.

A fixed script with each terminal executing a single program during each run, was chosen to facilitate the gathering of data. The benchmark programs were written in such a way as to restart the indicated operation upon each completion. This provided a continuous sample of each program from which the desired times could be obtained. Thus, to vary the load, the only requirement was to interrupt the operation of the selected terminal and change the program type to whatever one was desired.

V. MEASUREMENT TECHNIQUES

Before describing the measurement techniques, two types of performance measurements will be defined. The primary performance measurements are defined as the computer performance is observed by the users, such as turn-around times by priorities or job classes, throughputs in jobs per hour, and terminal response times. Primary performance measurements are useful in measuring the actual computer operation but are not very useful in improving performance. On the other hand, the secondary performance measurements are defined as the utilizations of computer resources, such as CPU, memory, channel, disks, monitor programs and compilers. These measurements are very useful, if not essential, in improving computer performance.

The primary performance measurements were obtained by observing the response times at the terminals and the throughput. The benchmark programs were written to give the real time at the commencement of a compilation and at the completion. From these figures the throughput for each job type was calculated, and a throughput factor was defined as follows:

$$TP_i = SS/(RD * NT_i)$$

where

SS = Sample size (number completed jobs)

RD = Run duration

NT_i = Number of terminals running program type i

This figure indicates the job completions per minute.

The secondary performance measurements for determining system efficiency were obtained by observing the time spent by the CPU in

problem state, wait state, supervisor state, and overhead, and the paging requirements. Fortunately a program called "MEASURE" was obtained from Mr. Stuart Wecker of the State University of New York, Stony Brook, New York, that provided just this information for CP. The program provided for various readings at different intervals and was run continuously while the benchmark programs were being run. Examples of the type of information provided for each interval of time are: CPU problem time, supervisor time, overhead time, pages read, pages swapped (written on backup storage) and pages stolen (removed from core while the user is in one of the active queues). In addition to the system totals, each user was monitored for the same information. From these observations, the percentage of CPU time spent in problem state and the paging rates were used as indications of the load on the system. No such measuring program was available for TSS, although a very complex data gathering program called SIP [Deniston, 1969] is now available from IBM.

VI. BENCHMARK DEVELOPMENT

The initial test was designed with the idea of constant growth in mind so that ultimately a sophisticated, valid benchmark would emerge which could be used with confidence. After each test was run and analyzed, the test was modified to meet requirements for the next test or to amplify information revealed by the analysis or simply to cull out weak points in the previous test. In this way the tests were improved by eliminating much of the wasted time in each succeeding run and by making the tests more representative of the load under actual operating conditions.

Three basic changes were made to the scripts upon completion of the first test:

1. The edit programs were changed to reflect think time of the user.

2. The ratio of edit-to-run programs was changed from 5:1 to 2:1.

3. The job mix was changed at 15 minute intervals with the terminal load varied from 12 to 22 users.

At this stage the benchmark programs were all finalized with the exception of EDIT. However, the test conditions for the third test were changed to reflect a heavier load and the run duration was lengthened to provide for a steadier load condition. To obtain a heavier load all edit programs were changed and the ratio of edit-to-run programs was further decreased from 2:1 to as low as 1:5. In the previous scripts, the editor was called only once at the commencement of the run and the attention interrupt was used to start and stop editing. Since TSS filed each edit command upon completion, the edit programs for CP

were changed to agree with the editing operation in TSS and to more accurately reflect a typical user's editing. The change was made by designing the exec routine which called the editor, performed three edit functions and filed the program. Finally, a program requiring heavy paging was inserted into the benchmark to observe its effect on the system.

The programs comprising the final benchmark were as follows:

1. FORTRAN - Fortran compilation.
2. FORTEX - Fortran execution.
3. PLISM - Medium sized PL/1 compilation
4. PLILG - Large PL/1 compilation
5. EDIT - Exec routine for editing a simple program and filing the results.
6. PAGE - Fortran program which executes a large matrix multiplication.

VII. DISCUSSION OF RESULTS

As mentioned in the preceding section a series of tests were conducted to accomplish the purposes of this thesis. The first three correspond to the stages in the development of the benchmark and offer a comparison of the two systems. Test four was concerned with the overloading and recovery from overload, while test five was intended to show that the results obtained by using fixed scripts would be essentially the same as those obtained by using mixed scripts. Tests four and five were not completed for TSS as intended because other demands for the computer made it impossible to schedule dedicated computer time for test runs.

A. TEST ONE: INITIAL TESTING WITH A LIGHT LOAD

This was a preliminary test in which the initial benchmark programs were tested on CP, as a result, little meaningful data was obtained and no comparison with TSS was made. The characteristics of the test were as shown in Table I.

Table I. Test Characteristics for Test 1

PROGRAM	RUN 1	RUN 2	RUN 3
EDIT	2	7	7
PLILG	1	3	2
FORTRAN	0	0	1
FORTEX	1	4	4
TOTAL	4	15	16

Although the results of the first test were not considered very significant because of relatively few users, problems in conducting the test, and other reasons, some results--namely average problem time and paging rates--are given in Figures 1-1[1] and 1-2 for information and to relate this test to the others.

B. TEST TWO: INTERMEDIATE LOAD

Test two was conducted with basically a 2:1 edit-to-run ratio and the job mix was changed at 15 minute intervals. This test was intended as an intermediate load comparison of the two systems. Table II gives the complete characteristics of the test that was run on CP. Four of the runs were compared with TSS to give an indication of the relative performances of the two systems although the runs did not constitute exactly the same load.

Table II. Test Characteristics Test 2

PROGRAM	RUN 1		RUN 2	RUN 3		RUN 4		RUN 5	
	CP	TSS	CP	CP	TSS	CP	TSS	CP	TSS
PLILG	0	1	0	1	1	1	1	1	1
FORTRAN	1	1	1	1	1	3	3	1	3
FORTEX	3	2	2	3	3	3	3	4	4
EDIT	8	8	10	12	12	14	14	15	16
PLISM	0	0	0	0	1	0	0	1	0
TOTAL	12	12	13	17	18	21	21	22	24

The problem time percentage and paging requirements for CP are given in Figures 2-1 and 2-2. These demonstrate that the system was

[1] The first part of the Figure number refers to the test number.

Figure 1-1. Average Problem Time

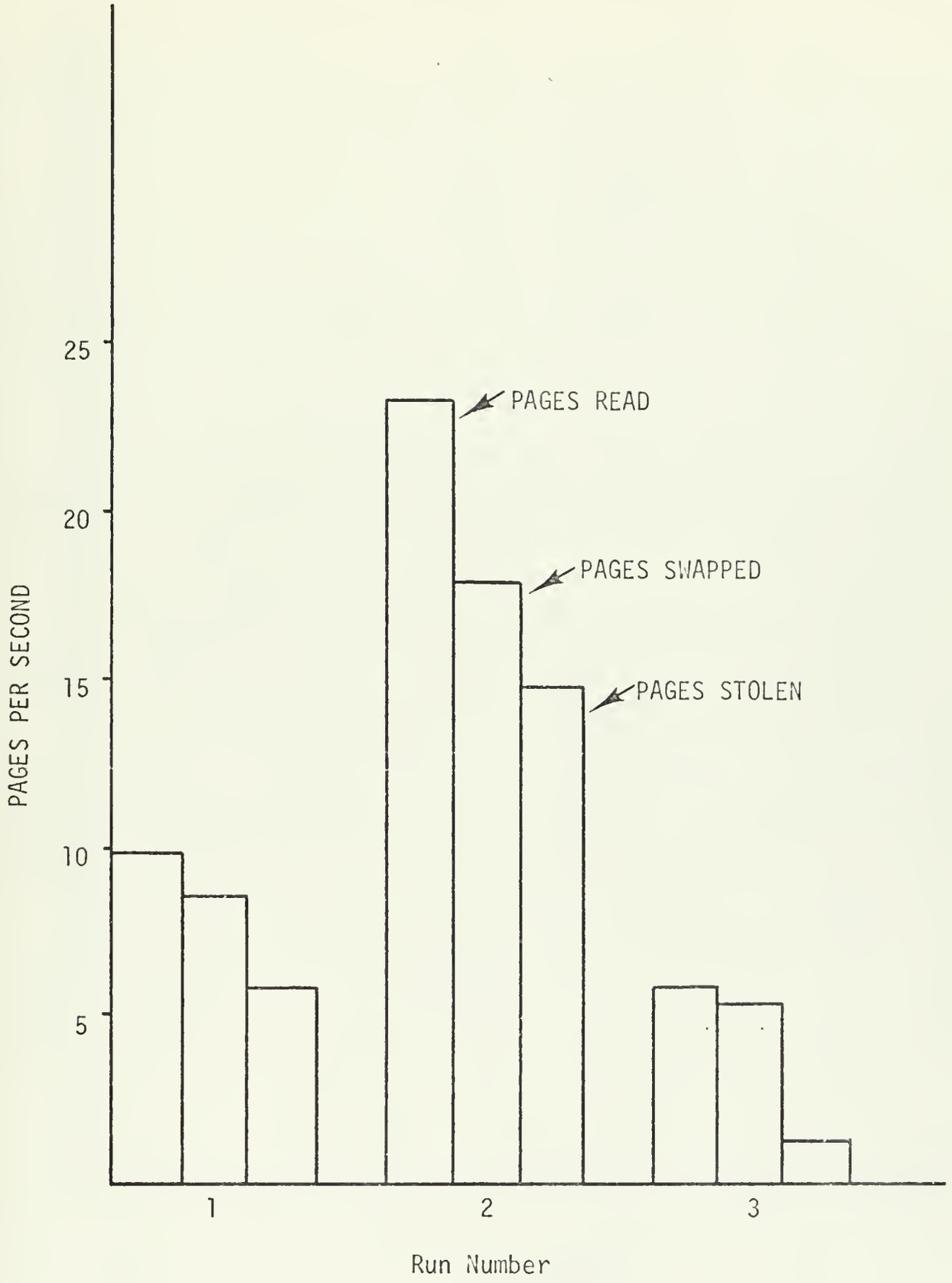

Figure 1-2. Paging Rates

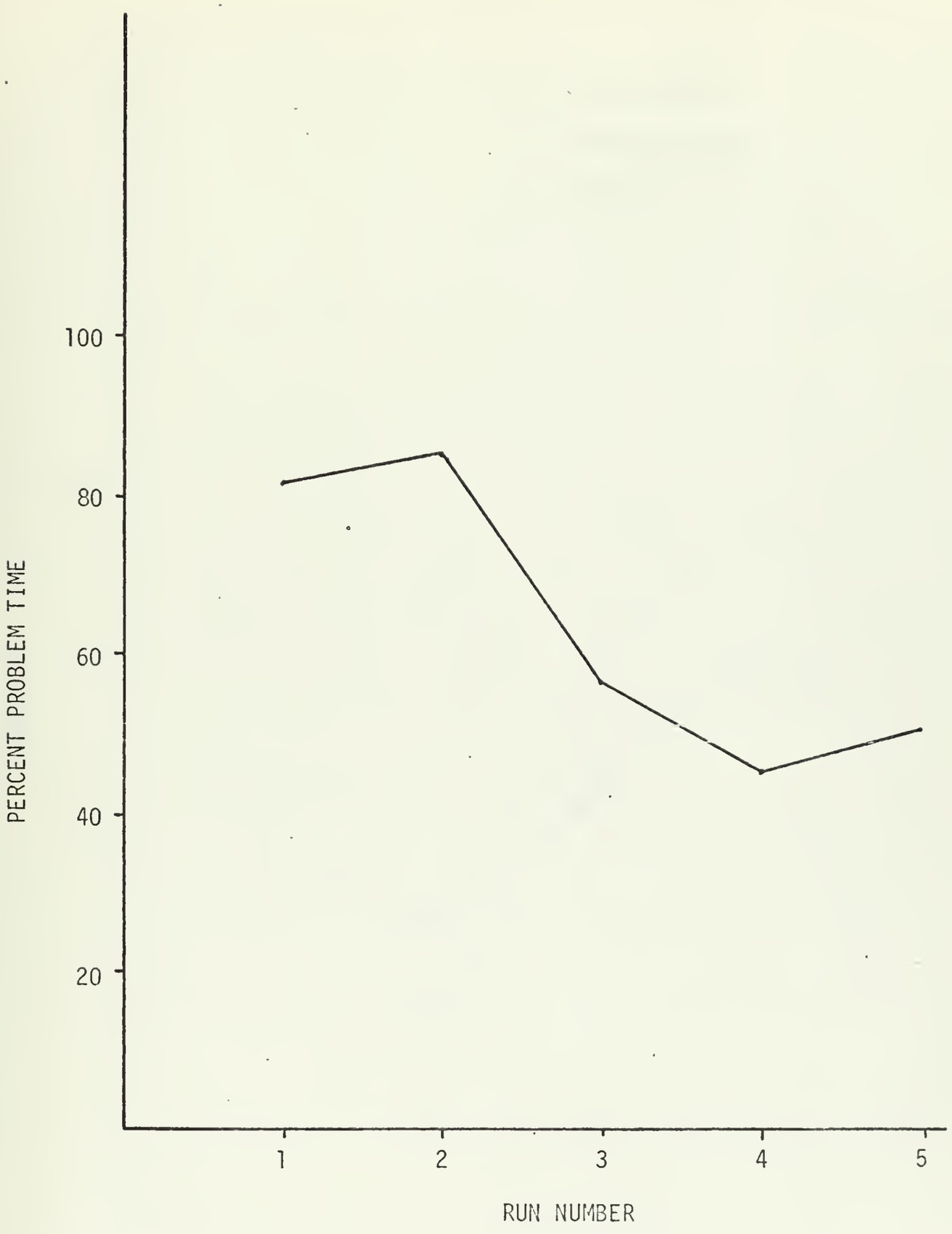

Figure 2-1. Average Problem Time.

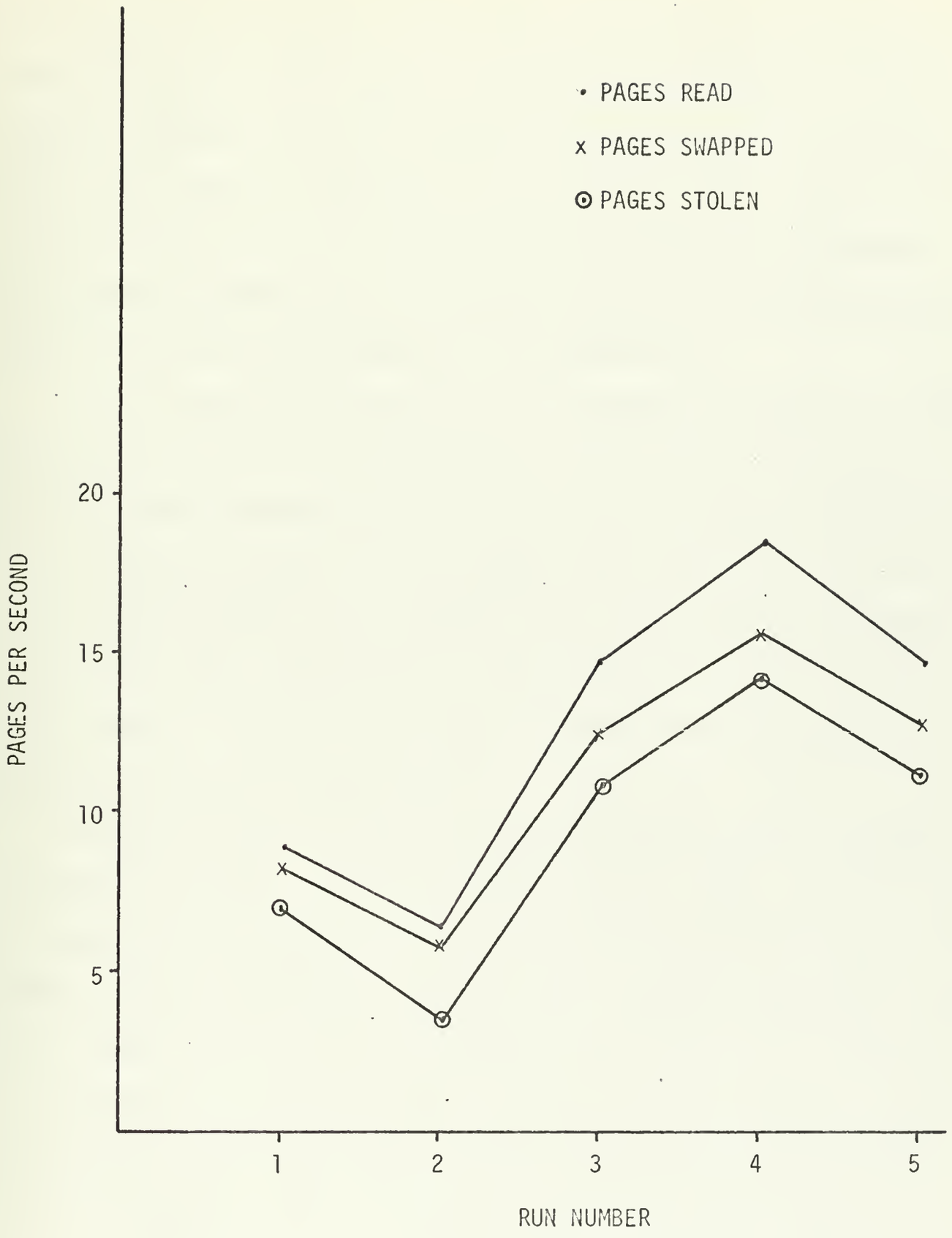

Figure 2-2. Paging Rate.

operating fairly efficiently and that the performance had not been degraded by overloading the system. The two primary indicators for performance degradation were the problem time percentage and the response time for the large PL/1 compilation. A problem time of 50 percent and a PLILG response time of five to ten minutes indicated an intermediate load, whereas a problem time of less than 20 percent or a PLILG response time of 30 minutes would indicate an overloaded system.

It was intended to use the five runs to gradually increase the load by increasing the number of terminals but it can be seen from Figures 2-1 and 2-2 that the variation in the load was not as great as desired. From observing the problem time and paging requirements, the load during runs one and two proved to be essentially the same, while the same holds true for runs three, four and five. This would indicate that the additional users and PLILG were the primary factors in reducing the problem time and increasing the paging load.

The response times of PLILG and FORTRAN for both systems are given in Figures 2-3 and 2-4. Run four provided the only direct comparison and showed that CP had the best response times for both PLILG and FORTRAN. Only two PLISM jobs were run and the response time for CP with 22 users was 2:27 and for TSS with 18 users it was 3:11. Figure 2-5 shows that the throughput factor for PLILG was almost the same for both systems, while it was better for CP in three of the four FORTRAN runs. Although the runs were slightly different and these graphs may not be completely accurate reflections, they do indicate that perhaps CP performed just as well if not better than TSS.

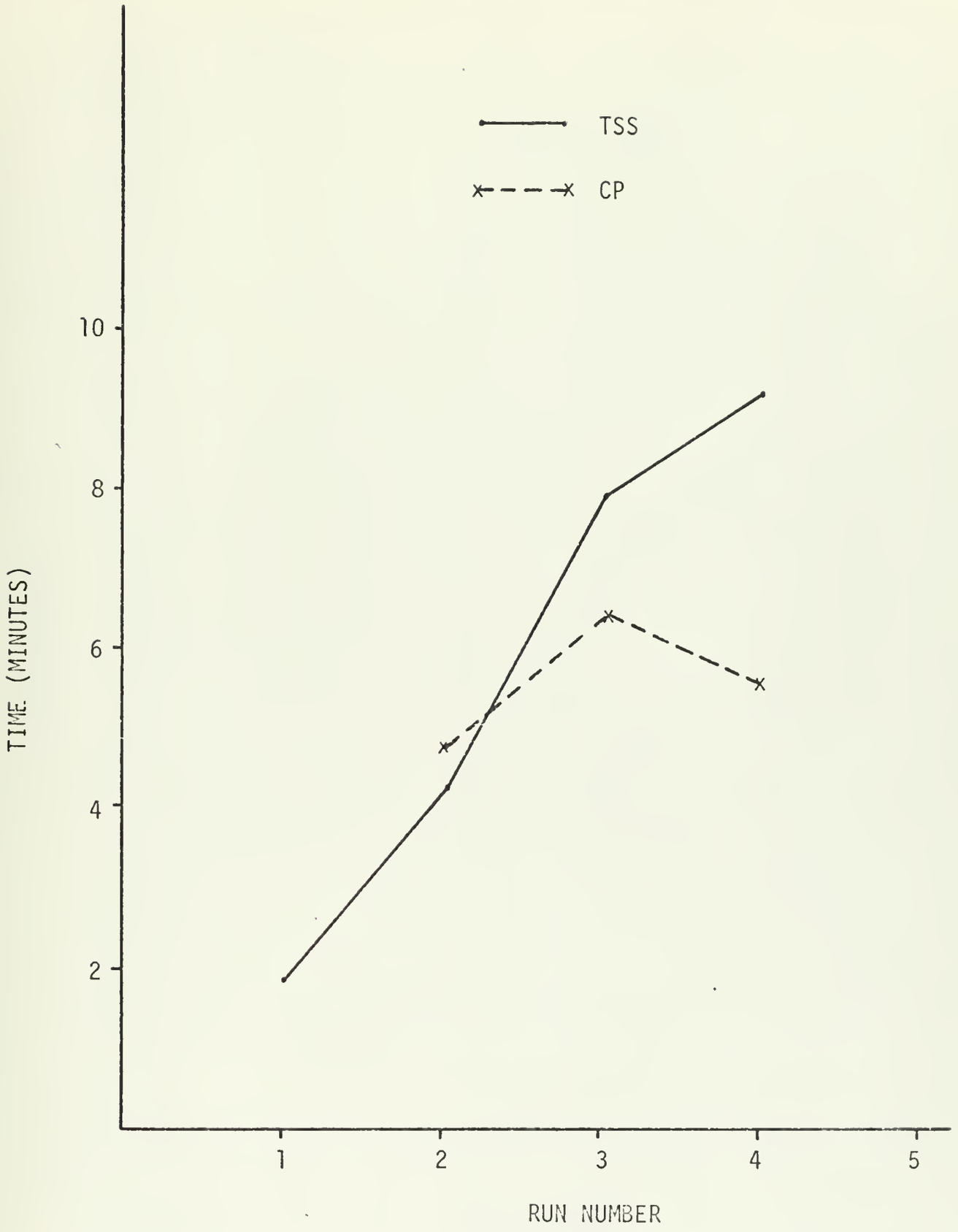

Figure 2-3. PLILG Response Time.

Figure 2-4. FORTRAN Response Time

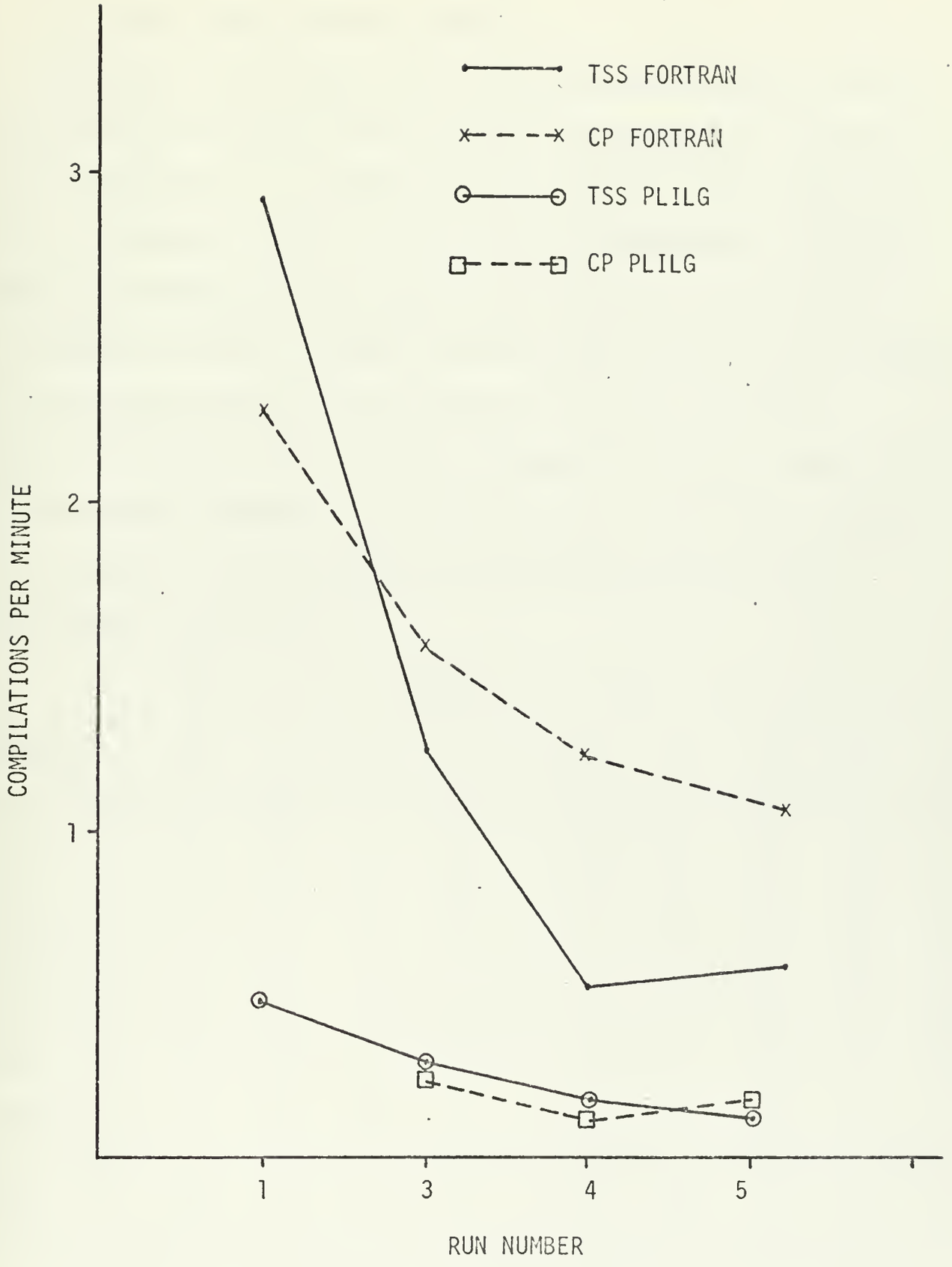

Figure 2-5. Throughput Factor

C. TEST THREE: HEAVY OVERLOAD CONDITION

Since one of the primary aims of the tests was to compare TSS and CP under heavy load conditions, test three was designed to increase the load on the system, by increasing the paging demands while holding the number of terminals constant at 24. This test offered the only direct comparison with TSS and consisted of five comparison runs with a CP configuration of 256K of storage. An additional run was made on CP with 512K which consisted of the same script as run one. As before, a summary of the test characteristics is shown in Table III. Since the run duration was increased from 15 to 30 minutes, a more nearly steady state condition was achieved and the data gathered was the most significant of the three tests.

Table III. Test Characteristics for Test 3

PROGRAM	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
PLILG	4	2	2	2	2
PLISM	3	3	3	3	3
FORTRAN	7	5	5	5	5
EDIT	4	12	10	8	6
PAGE	0	0	2	4	6
FORTEX	6	2	2	2	2
TOTAL	24	24	24	24	24

It can be seen from Figure 3-1 and 3-2 that the load on CP was much higher than on previous tests. The percent of time the CPU spent in problem state was much lower and the paging requirements much higher than test two, which indicates the desired overload condition.

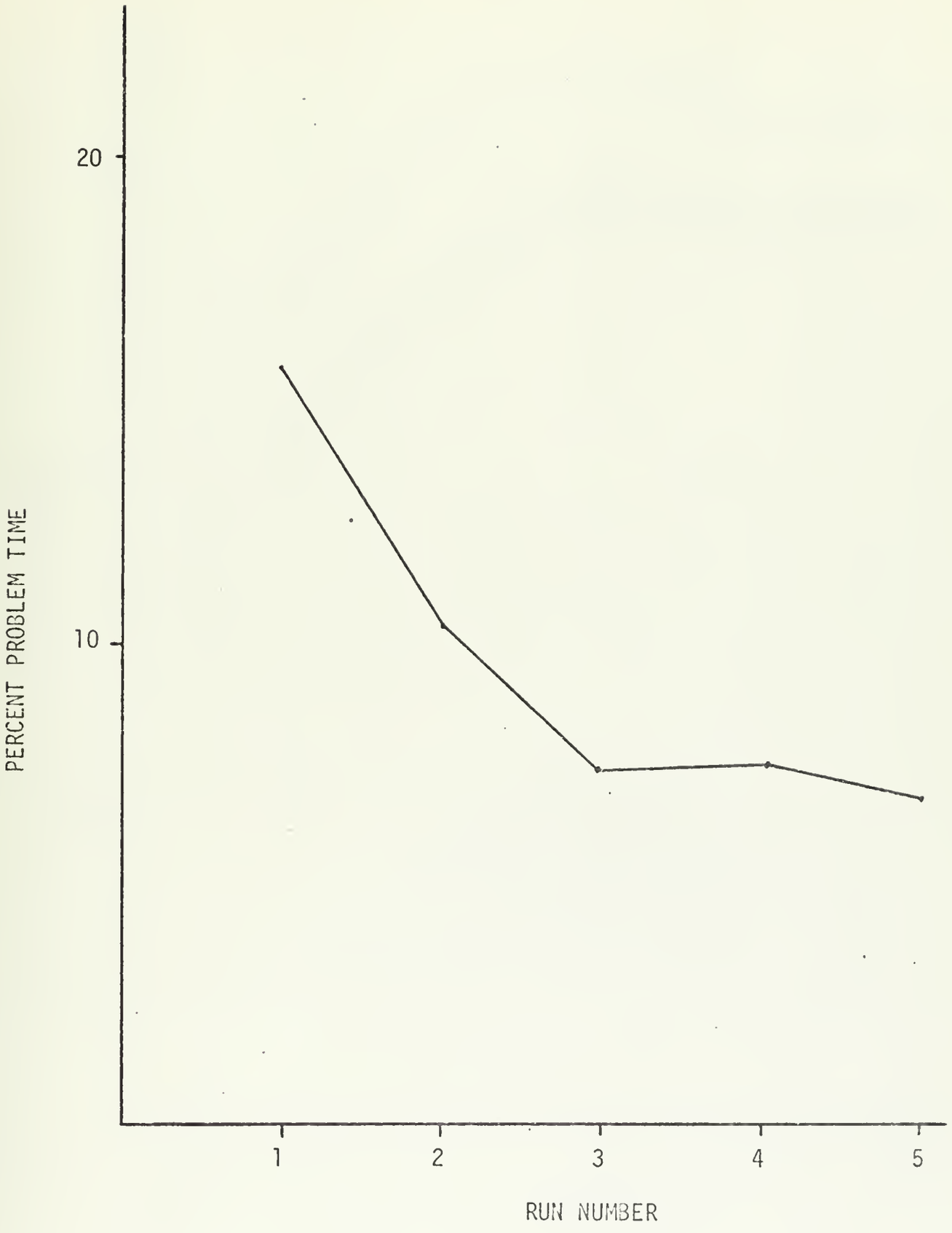

Figure 3-1. Average Problem Time

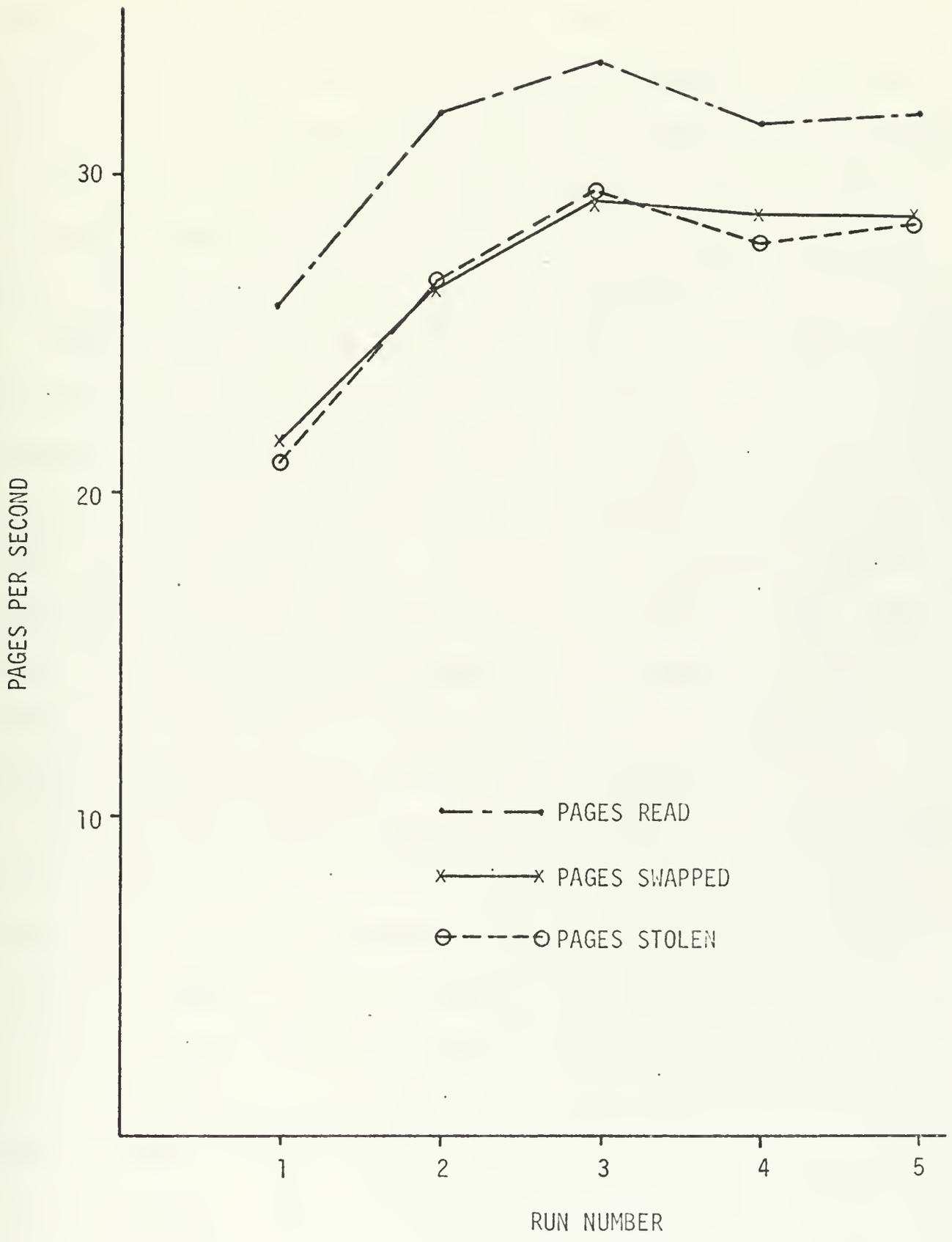

Figure 3-2. Paging Rates.

Although a heavier load is evident it was somewhat different than expected. The intent of the test was to have the heaviest load on run one, cut back to the lightest load on run two and then gradually increase it again. It can be seen however, that this was not the case. Instead of being the heaviest load, run one was actually the lightest and the load for the remaining four runs was almost the same.

Another item of surprise was that the paging rates of runs two through five were almost the same. The program PAGE was inserted to increase the paging requirements and thereby the load of the system. To observe the effects of this increase in paging, the load was changed by the replacement of EDIT users with PAGE users. Instead of gradually increasing, the page rates remained constant for these four runs. The reasons for this unexpected occurrance will be discussed in detail in a later section. Although the load was not exactly as desired, both computer systems were operating under the same conditions.

Since no data gathering program, such as MEASURE, was available for TSS to collect problem time and paging statistics the only means of direct comparisons were the response time and throughput statistics. Figures 3-3 through 3-5 give the comparison of the response times of the three different compilation programs under CP and TSS. Since there were no heavy paging users during the first run, the times were approximately the same. For the remaining runs the times were significantly different, however, since the edit and paging loads were increased. The difference in response times for PLISM and FORTRAN were not as great as for PLILG, with CP having smaller times for PLISM and TSS smaller for FORTRAN.

Figure 3-6 offers a comparison of the throughput for the two systems by displaying the throughput factors for each run. This figures verifies

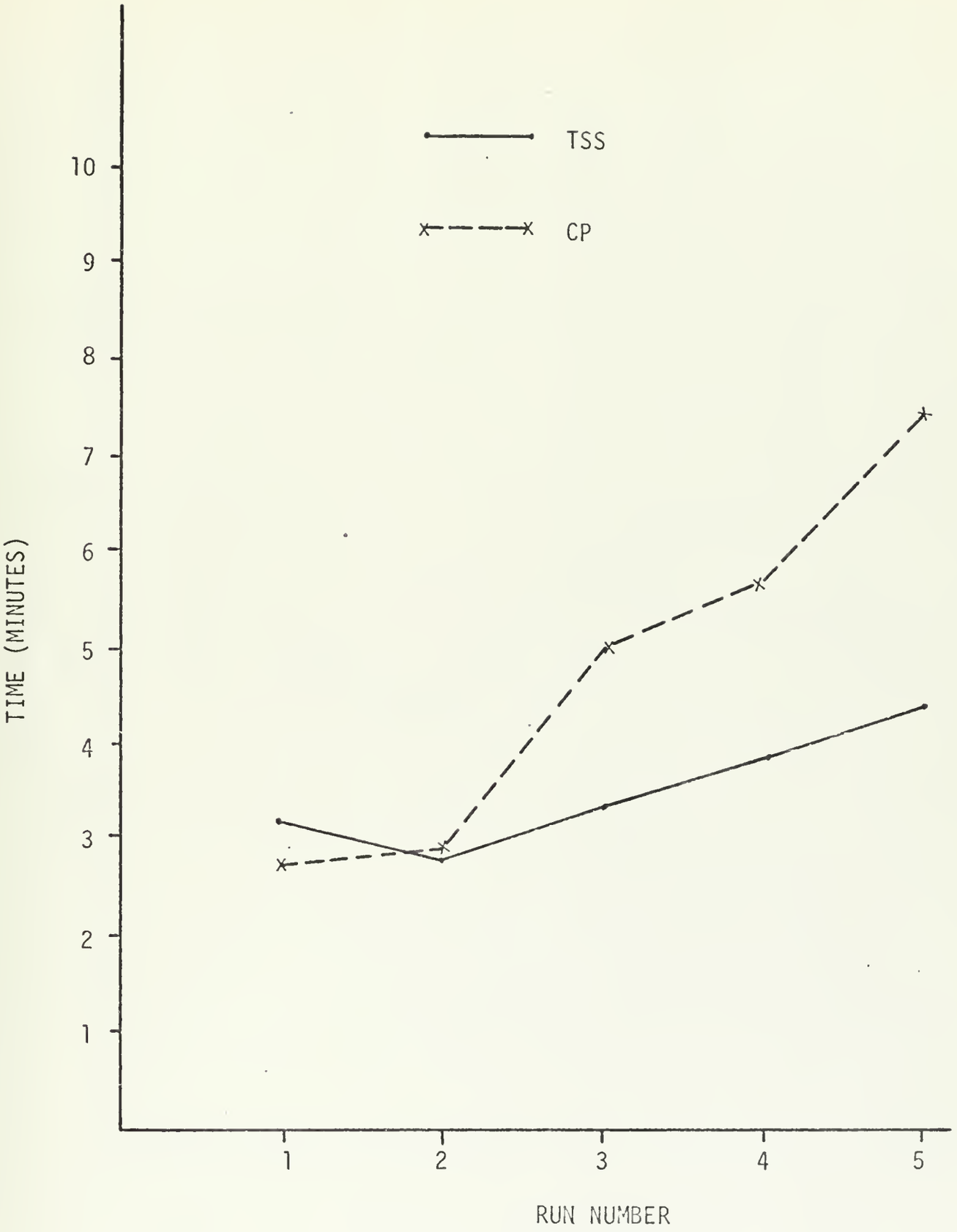

Figure 3-3. FORTRAN Response Time.

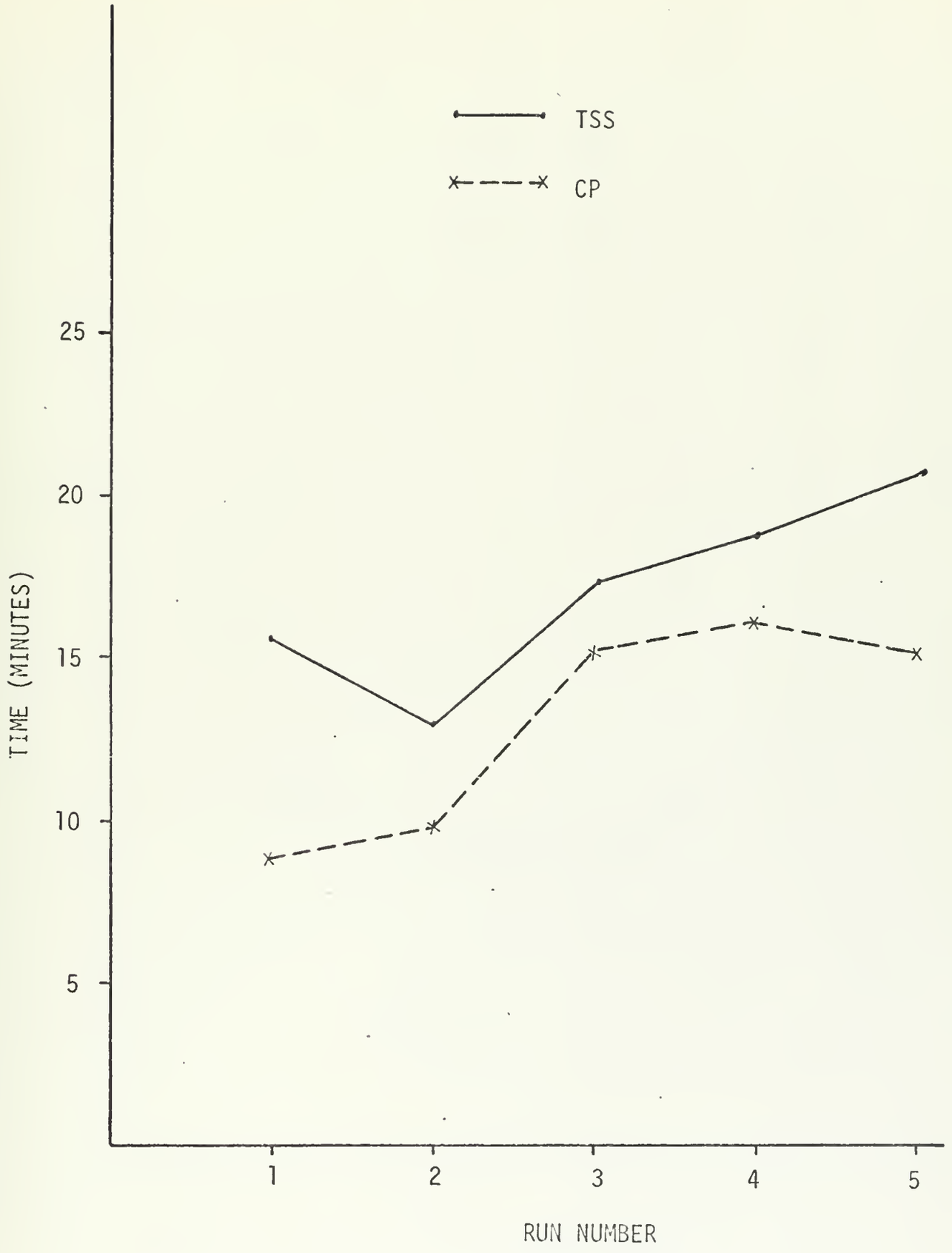

Figure 3-4. PLIGSM Response Time.

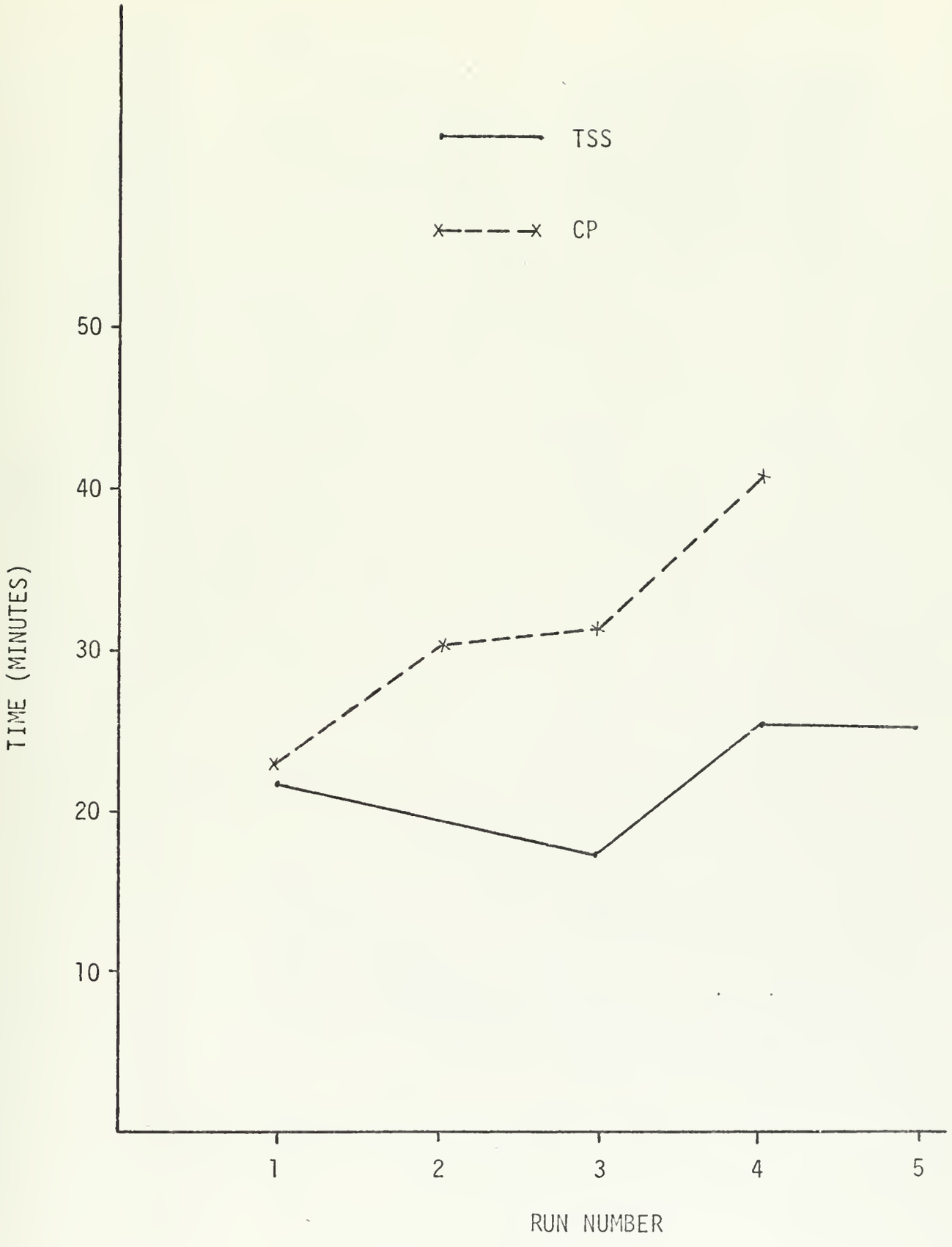

Figure 3-5. PLILG Response Time.

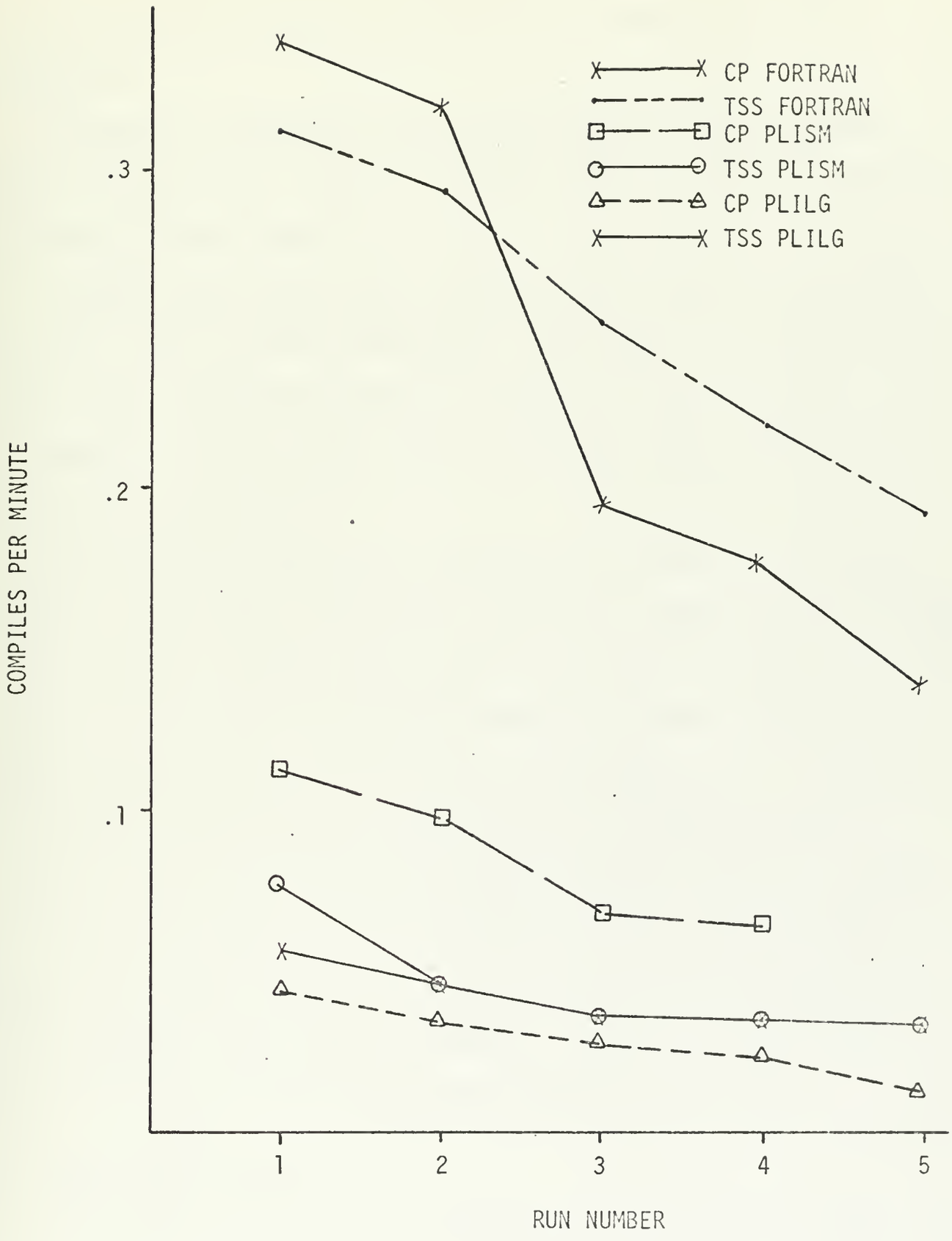

Figure 3-6. Throughput Factor

that run one was the lightest load and that it steadily increased afterwards.

Although the throughput varied with the type of job, the general observation was that it was better for CP during the first two runs and for TSS on the last three.

The next three figures offer a comparison of TSS and CP with 256K and 512K. They show that the additional 256K bytes of core memory provided a very significant increase in performance level of CP over TSS, even though CP still has 256K bytes less memory, slower disks and one less processor than TSS.

The Effective Progress Rate was used in this test as an additional indicator of the load on the system. This rate is defined [Lasser, 1969] as the time required for a program to run stand alone divided by the time required to run under the given load condition. These rates for the three compilation programs for CP are given in Figure 3-10. If the sum of effective progress rates for all programs of a particular type currently being executed is one, then the computer is operating as well as a serial processing computer. The amount over one indicates the degree to which the performance is improved by overlapping the use of various resources under multiprogramming. Effective progress rates much less than one indicate the system is heavily overloaded.

D. HEAVY PAGING EFFECT

The paging results that were obtained in test three were rather perplexing in that the paging rates were essentially constant over the last four runs while each run contained two more PAGE users than the previous. This was first contributed to the revised EDIT program by assuming that the more frequent editor calls required more paging, with the result

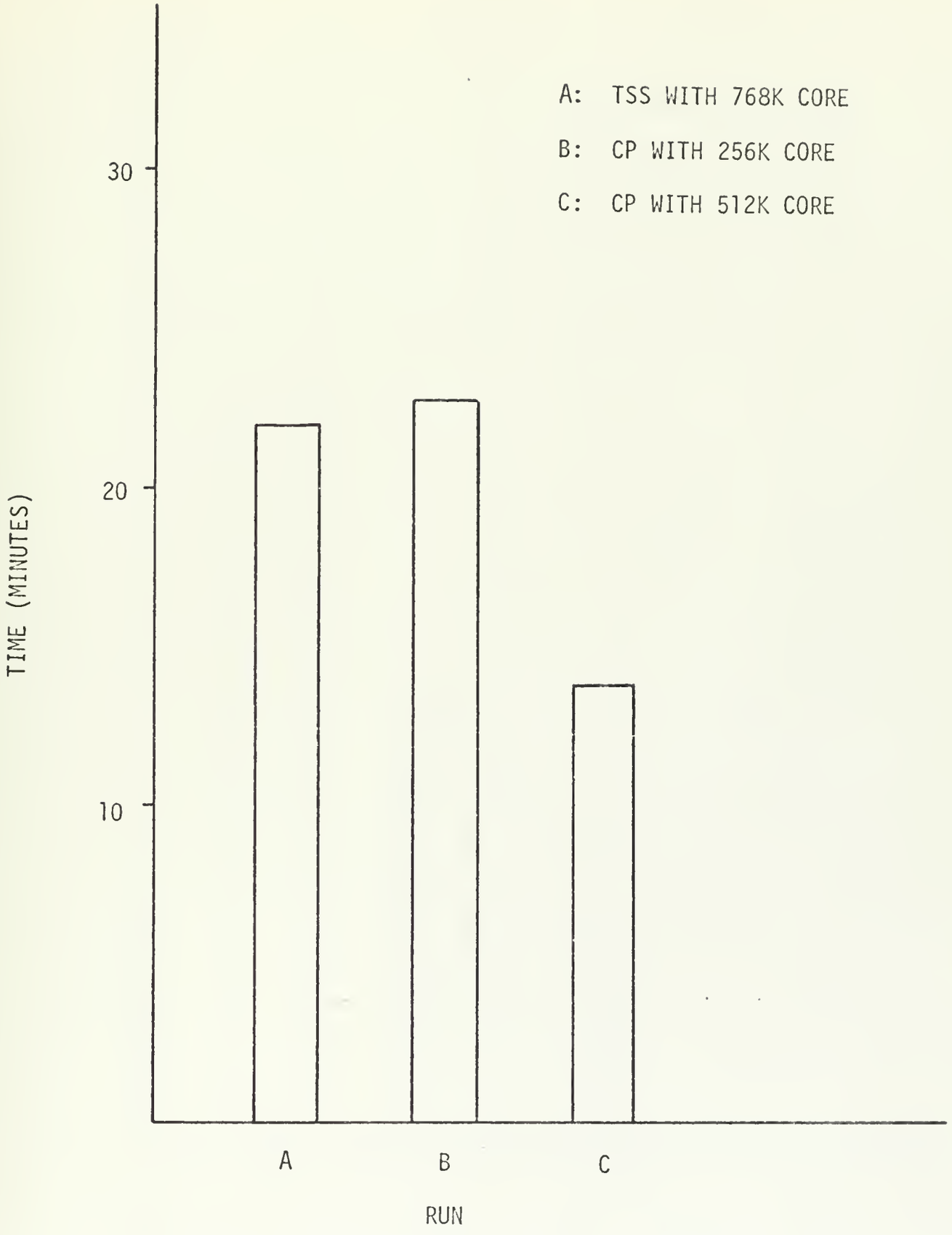

Figure 3-7. PLILG Response Comparison.

Figure 3-8. PLISM Response Comparison.

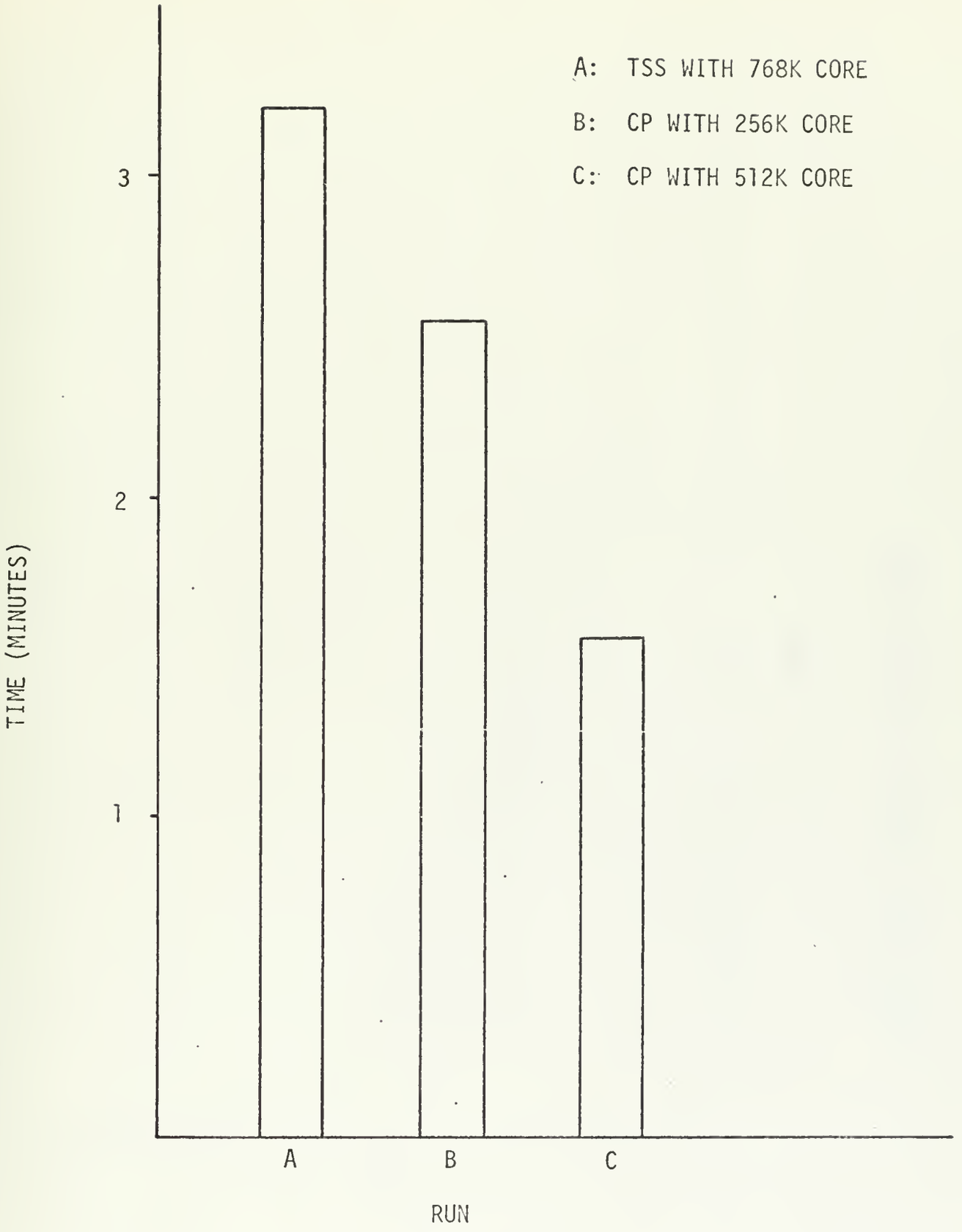

Figure 3-9. FORTRAN Response Comparison.

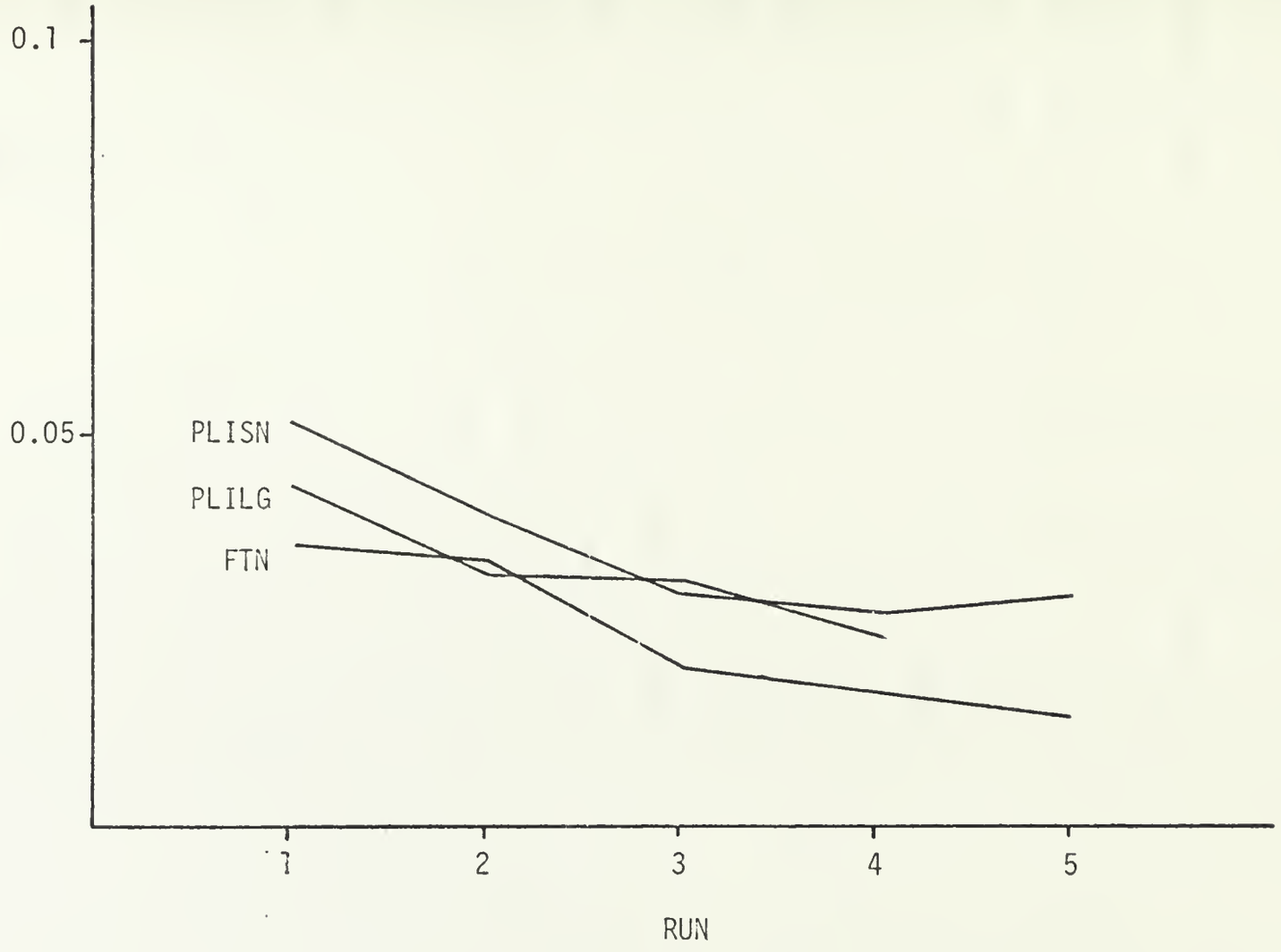

Figure 3-10. Effective Progress Rates

that the combination of EDIT and PAGE users offered the same load. Upon further investigation, it was concluded that this was not the case and that the programs did not present the same load to the system. EDIT did in fact increase the paging requirements of those users over the proceeding tests, but the total paging requirements were much less than that of PAGE. This can be seen from Figure 3-11 which gives the paging rates of each individual program type. It was thought that the addition of PAGE programs would increase the paging rates proportionally, but this was not indicated by the results. Response time and throughput indicate that the load increases steadily as the number of PAGE programs increased. The question is, "Why did the overall paging rates remain constant while the load was being increased?"

There were several related causes, but the basic idea is that PAGE placed such a load and overwhelming paging demand on the system that the maximum possible paging rate was achieved. It is assumed that this paging rate is limited by channel capacity and indicates that the system was heavily overloaded. Although there was little variation in the paging rates, the compilation and execution programs were still affected because the paging rates for PLILG and PLISM dropped significantly once PAGE was started. The requirements for PLILG and PLISM were still the same, but since PAGE was the dominant user, it increased the response times and decreased the throughput for these other two programs. Essentially the conclusion is that PAGE placed such a load on the system that it overshadowed the remaining programs and pushed the paging rates to a maximum. More tests would have to be conducted to verify that this is a maximum possible paging rate.

Although paging requirements were a key factor in determining how much utilization the CPU had in the problem state, it must be remembered

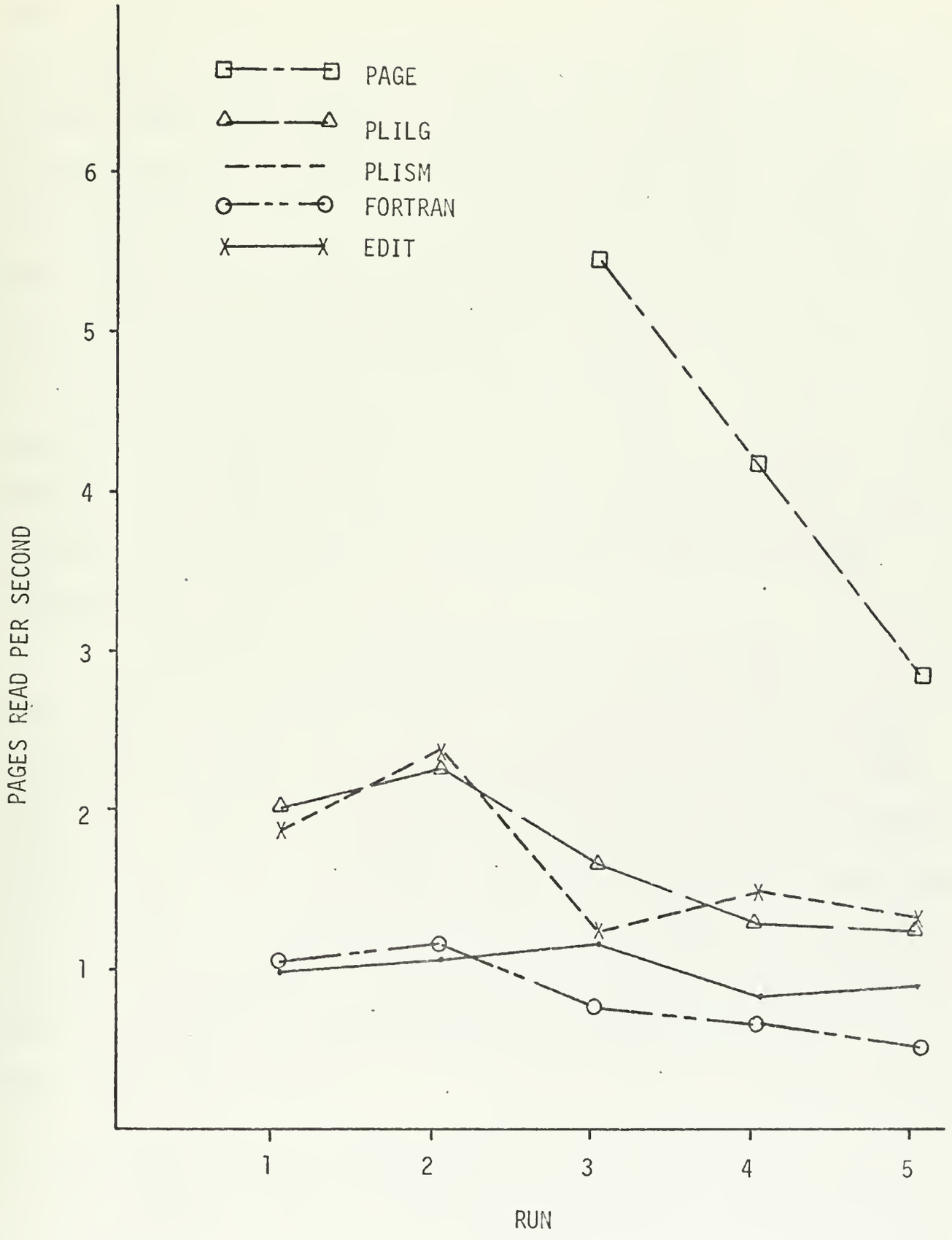

Figure 3-11. Program Paging Rates.

that the channel activity, especially in the multiplexor channel to the terminals, also affected CPU utilization. Even though an inverse relationship between paging and CPU utilization was established, no such relation seemed to exist where channel I/O was concerned. Figures 3-12 and 3-13 show the relationships between average problem time and paging rates. To convert the paging rates to a comparable scale with problem time an arbitrary factor was chosen. This factor was chosen with 100 as a reference and arbitrarily assigning that figure to the maximum paging rate achieved to obtain the factor, α. Thus α is related to the time to read a page such that the maximum measured paging rate produces 100 percent channel capacity. α was determined from the following relationship:

$$\alpha = 100/(\text{pages read rate} + \text{pages swapped rate})$$

Thus, α was calculated to be 1.58 for a maximum paging rate of 62.9, and then used to obtain the paging requirement indicator for the rest of the runs. Figure 3-13 shows the direct relationship between problem time and paging rates. The points include problem time percentages from all three tests. A graph of this nature could be used to ascertain the threshold of the paging rates for a particular problem time although many more observations would have to be made to accurately make a significant determination.

E. TEST FOUR: RECOVERY FROM OVERLOAD

The intention of test four was to start with an intermediate load (40-50 percent CPU utilization) and increase the load until an overload condition existed, then cut back to the intermediate load and observe the effect on the system in terms of the time required to recover from overload. The test was only conducted on CP since no means were available

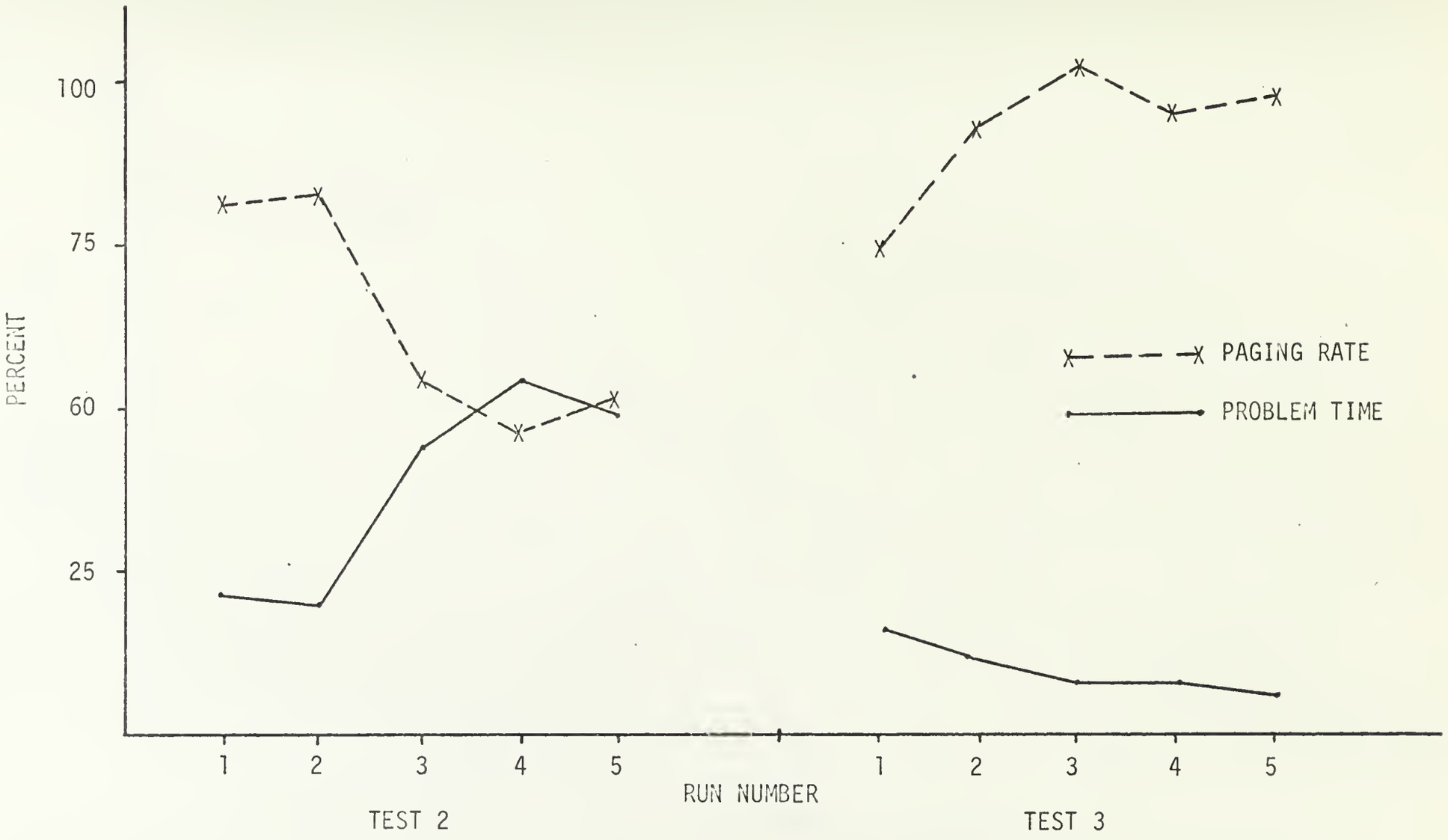

Figure 3-12. Channel Utilization Factor.

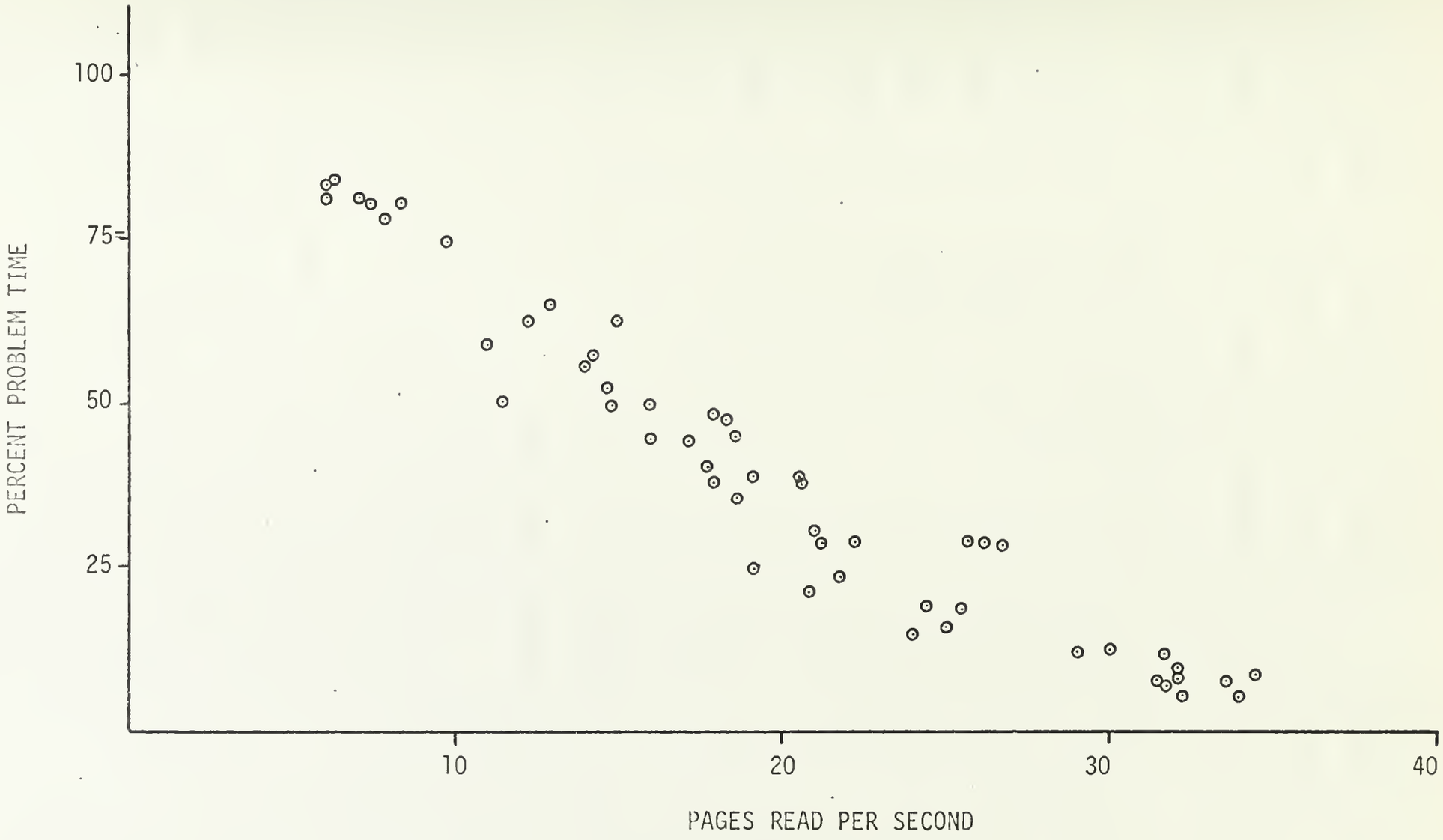

Figure 3-13. Problem Time versus Paging.

in TSS to observe the necessary information on resource utilization. Table IV shows the characteristics of each run of the test, with run two being the intermediate load level and runs three through six the increments by which the load was increased. Run seven was configured exactly as run two to observe the recovery time.

Table IV. Test Characteristics - Test 4

PROGRAM	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	RUN 6	RUN 7
PLILG	1	1	4	7	13	15	1
PLISM	1	1	1	1	2	2	1
FORTRAN	1	1	1	1	1	1	1
EDIT	14	13	10	7	0	0	13
FORTEX	5	6	6	6	6	4	6
TOTAL	22	22	22	22	22	22	22

From observations of past tests it was decided to increase the load by increasing the large PL/1 compilation and decreasing the EDIT programs, since PLILG was the dominant paging user. With the exception of runs five and six the number of other programs was kept fairly constant. This would allow for the observation of the effect of a heavy PL/1 load on the system. Figures 4-1 and 4-2 offer two separate views of the CPU utilization. The first shows a plot of the problem time percentages versus time and it can be seen that the recovery from overload was very short, as the CPU utilization returned to the same level as run two within four and one half minutes. The points plotted in Figure 4-1 were obtained from MEASURE and are the successive readings throughout the test rather than the run averages. Both problem time and roblem


60
50
40
30
20
PERCENT
PROBLEM TIME
PROBLEM + SUPERVISOR TIME
1
2
3
4
5
6
7
1
(TIME HOURS)
2
3

Figure 4-1.

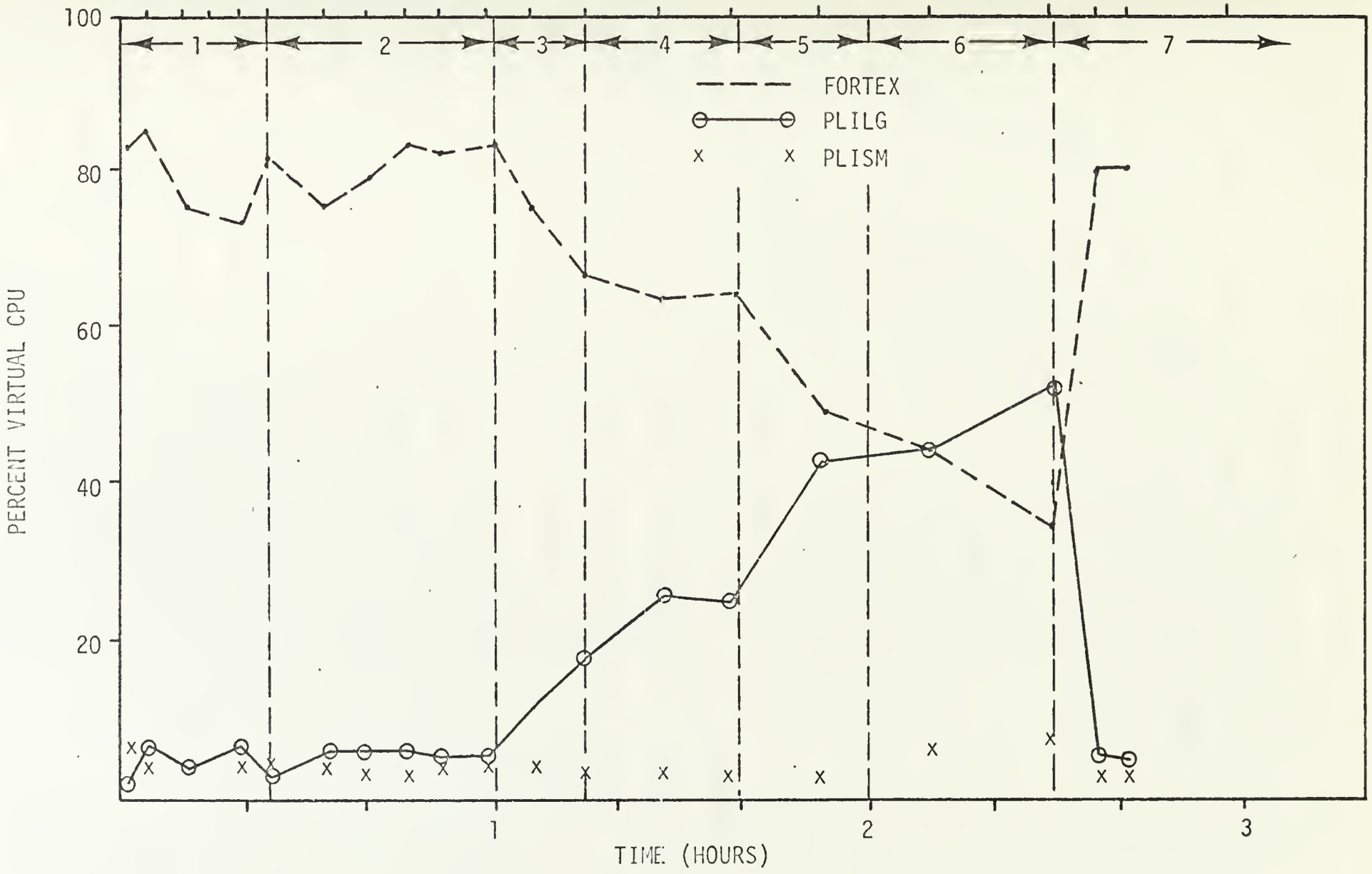
100
80
60
40
20
PERCENT VIRTUAL CPU
1
2
3
4
5
6
7
FORTEX
PLILG
PLISM
1
2
3
TIME (HOURS)

Figure 4-2.

time plus supervisor time are plotted. This shows that the percentage of time that the CPU spent in supervisor state was essentially constant at 12-15 percent for each run and that the heavier load does not effect this percentage. (Although not shown on the earlier tests, this was also found to be true for the first three tests. The percentages varied somewhat with the number of users on the system but not with the varying job mix for a constant user load.)

Figure 4-2 shows the breakdown in virtual CPU percentages by program type on the same scale. It can be seen from this that FORTEX had by far the largest percent of CPU utilization until the number of PLILG's were increased to over double that of FORTEX, during run five.

Figure 4-3 displays the paging rates and, as can be seen, there was not a large amount of variance in the runs. All other indications were that the load was steadily increasing and overload was being achieved. However, the constant paging rates appear to be attributed to the constant combination of EDIT and PLILG programs. This assumption is unsubstantiated but believed to be accurate. Whereas the constant paging rates of test three were attributed to the maximum channel capacity, the reasons for the constant rates in this case are not the same, since the maximum rate achieved in test three is approximately one-third greater than the maximum reached for this test.

Figure 4-4 gives the average response to the three different compilations and the average time required to complete one cycle on the FORTRAN execution (FORTEX). This cycle required 16 seconds of CPU time to complete. These give another indication of the load on the system. Although the load for this was not as heavy as that of test three, it still serves the purpose of observing the recovery as the system was considered to be very near the overload point according to the criteria

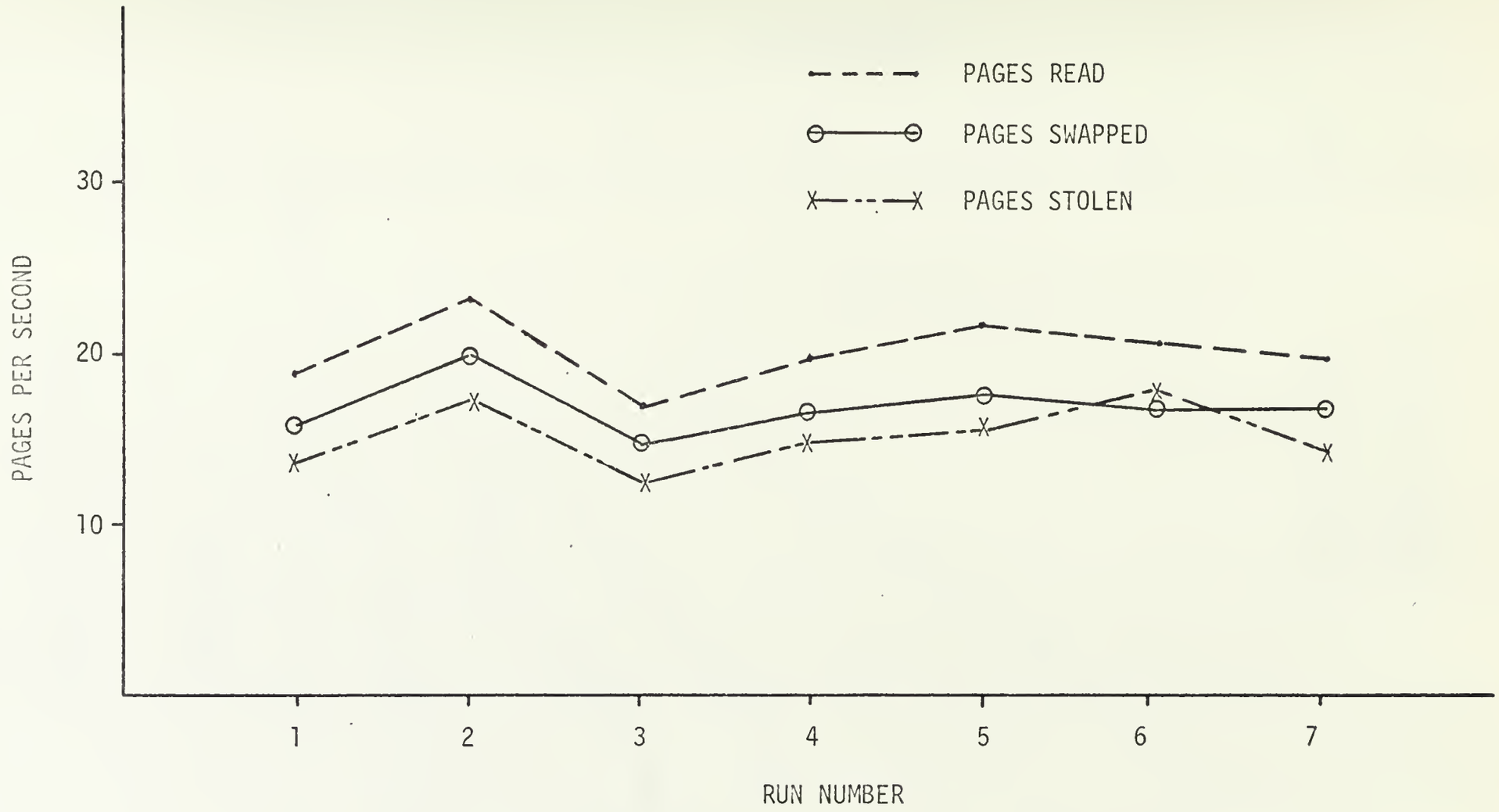

Figure 4-3. Paging Rates

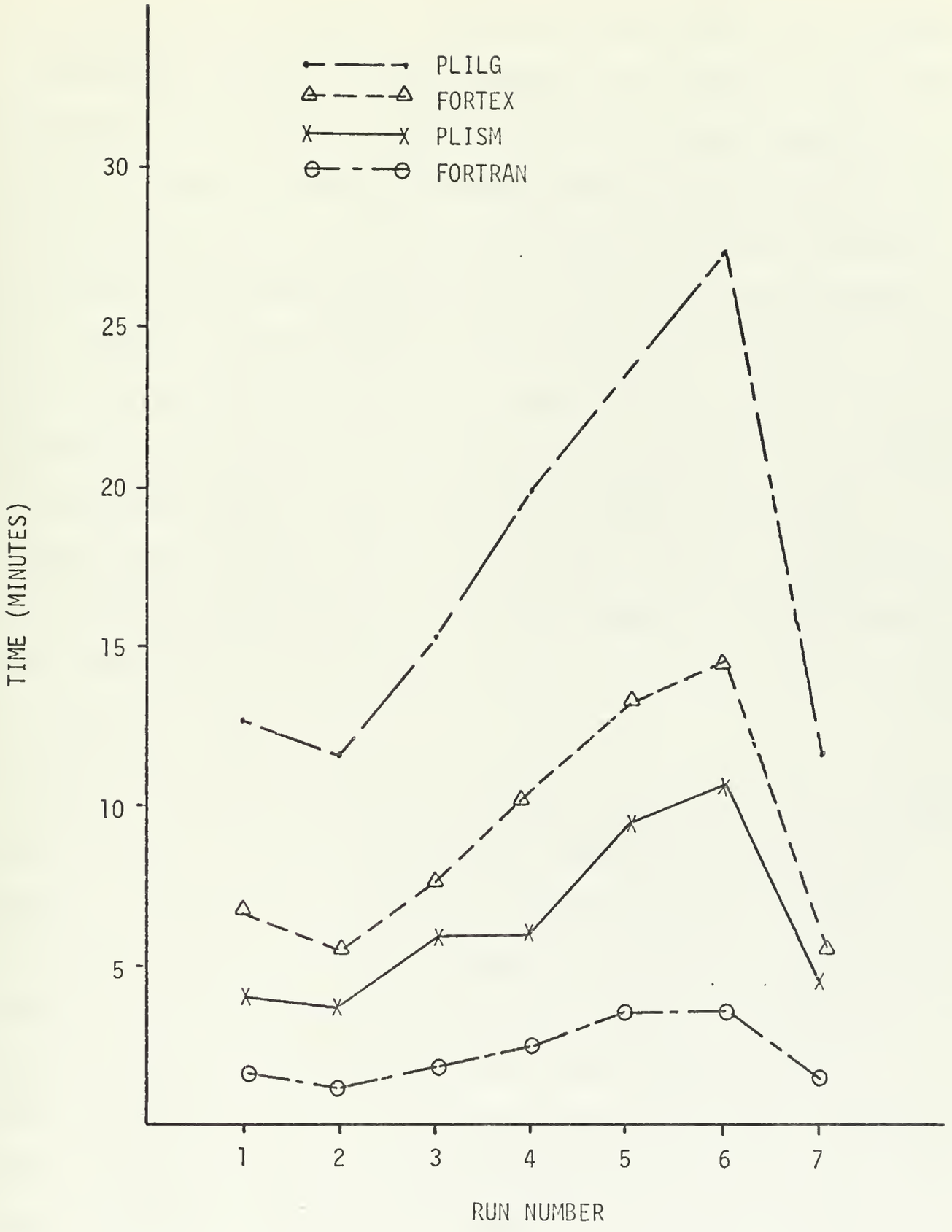

Figure 4-4. Response Times.

set forth in an earlier section. PAGE was not used in this test because of file transfer problems and this resulted in the somewhat lighter load than desired. The effective progress rates for the three compilation programs are given in Figure 4-5 and again give the same indication of the load and recovery run. It can be seen that as the load increased the progress rates became more nearly the same for the three programs, which indicates that the system was overloaded and no program was dominant since little work was getting accomplished on any job. This is borne out by Figure 4-6 which gives the throughput factor for this test. The throughput steadily decreased with the increase in the load on succeeding runs until the recovery run was made. On the recovery run it can be seen that the throughput returned approximately to that observed on the intermediate run (run two).

F. TEST FIVE - MIXED SCRIPTS

Test five was conducted to substantiate the fact that a fixed script on each terminal is a realistic method of loading the system for test purposes. An intermediate load run was chosen (run two of test four) and a mixed script was designed to compare the results with those of the previous test. Three separate scripts were written in an attempt to avoid the cycling problem whereby the same type of programs have a tendency to become synchronous in the requests for resources. The same number and types of programs were included in all three scripts. However, the order of execution was different. The average problem time percentages, paging rates and response times for the entire test were as follows:

Percent Problem Time	35
Pages read rate	20.7 per sec.

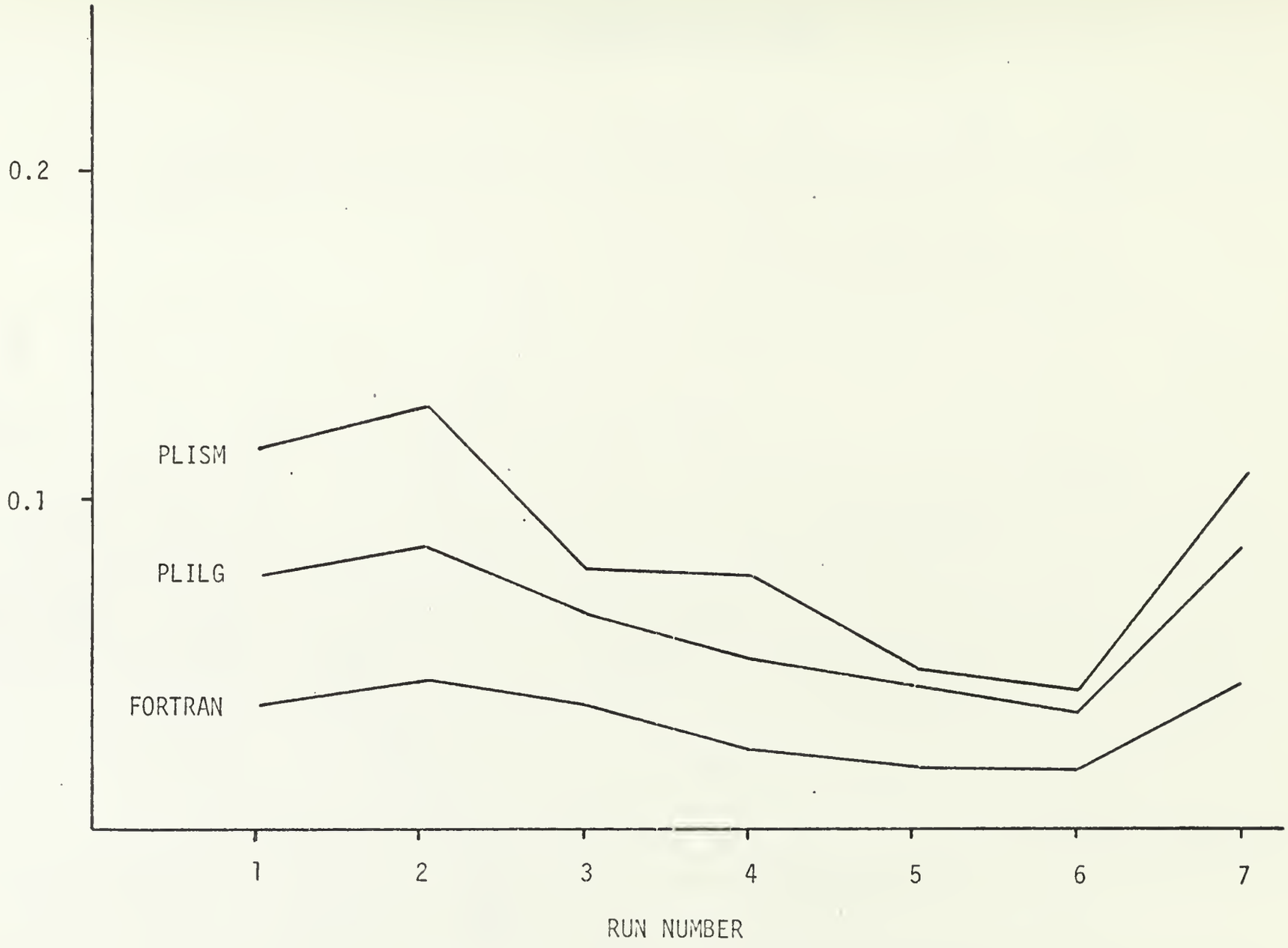

Figure 4-5. Effective Progress Rates.

Figure 4-6. Throughput Factor.

Pages swapped rate	16.8 per sec.
Pages stolen rate	17.8 per sec.
PLILG response	20.8 min.
PLISM response	10.2 min.
FORTRAN response	3.5 min.
FORTEX cycle	9.7 min.

Although the overall averages were practically the same as those of the fixed script test, there was a large amount of cycling present. This is to be expected since it is only logical that the compilations would vary depending on what other jobs were running at the same time. The PLILG response times varied from 12:10 to 45:34, PLISM from 4:15 to 27:34 and FORTRAN from 1:24 to 7:09. The completion of a FORTEX cycle varied from a low of 3:59 to a high of 23:01. The important fact, however, is that the overall averages are the same as for the identical run under a fixed script. This confirms that the procedure of using a fixed script is acceptable, because it shows that the system is not biased by running either method providing the tests are conducted for a sufficiently long time.

VIII. CONCLUSIONS AND RECOMMENDATIONS

The most obvious conclusion from these tests is that CP can compare with TSS over a wide range of load conditions even though CP requires significantly less computer resources. Thus, it is much better to run CP with one processor and one core box (256K bytes) and a batch system on the other processor and two core boxes rather than to run a dual processor 360/67 with three core boxes under TSS. Furthermore, it has also been concluded that CP with two core boxes will provide responses that are twice as fast as TSS running with three core boxes.

The original object of the study - to compare TSS and CP - was accomplished satisfactorily. The benchmarks that were derived in the research were adequate to compare the two systems over a wide range of load conditions. Although any benchmark is almost certain to be treated with criticism and skepticism, simply because of the wide variety of demands at different installations, it is considered essential that such benchmarks be derived so that the computer systems can be compared. Two types of comparisons were made. The primary performance measurements - response time and throughput - were the most useful in making the comparison, while the secondary performance measurements - utilization of system resources - were less reliable indications of the system load. It appeared that the inter-relation between problem times, paging requirements, and channel activities made it difficult to interpret the secondary performance measurements.

The two basic problems in any work of this kind is that of the proper test design and adequate data collection means. MEASURE is a step in the right direction but is available only for CP, and many more

software measurement programs are needed. These programs must be simple to analyze, easy to use, and not require many system resources.

The fixed script method of data collection provides the best means of gathering information about the performance from the user's criteria for several reasons. The first being that the fixed script allows a steady state condition to be obtained in much shorter times than the mixed scripts. Also the data obtained from a fixed script is easier to analyze since no searching through various program types is necessary. Furthermore, it was shown that both types of scripts produce essentially the same load conditions and performance.

An alternate method of determining which of the two systems is best suited for use at the Naval Postgraduate School would have been to use a hardware monitor to measure the system while it is running under actual operational conditions. However, this would probably require a minimum of one year - six months for each system - to collect sufficient data to make a fair comparison, considering the normal variation in user demands between academic quarters. This is not to infer that a hardware monitor would not be useful in benchmarking a system, since such a monitoring device, used in conjunction with a good benchmark, would be invaluable in comparing the two systems. It would then be possible to compare many more facets of resource usage from core utilization to channel load within the computer, and to do this in relatively short periods of time.

A few recommendations are set forth as guidelines for continuing this particular study. First of all a more detailed study into the relationship between CPU utilization, paging requirements and channel capacity is strongly urged. These three factors seem to have a strong bearing on the efficiency of the computer and the satisfactory performance to the user. It is known that they each affect the system

tremendously, but how much they are interrelated is only conjecture at this point. Rigorous tests using a hardware monitor is mandatory to explore as many combinations of these factors as is feasible.

A second recommendation would be to try and make the scripts as realistic and true to actual usage at this installation as possible. This would mean using more languages and a greater variety of programs than was attempted in this research. Additionally a good accurate sampling of actual user loads over a long period of time is required, although it is realized that this is a very frustrating problem at an educational institution.

BIBLIOGRAPHY

1. Calingaert, Peter, "System Performance Evaluation Survey and Appraisal", Communication of the ACM, v. 10, no. 1, January 1967, p.12-18.

2. Deniston, W.R., "SIPE: A TSS/360 Softward Measurement Technique", Proceedings of the 24th National ACM Conference, 1969, p.69.

3. Doherty, W.J., "Scheduling TSS/360 for Responsiveness", Proceedings of the Fall Joint Computer Conference, 1970, p.97-111.

4. Estrin, G. and Kleinrock, L., "Measures, Models and Measurements for Time-Shared Computer Utilities", Proceedings - ACM National Meeting, 1967, p.85-67.

5. Karush, A.D., "Two Approaches for Measuring the Performance of Time-Sharing Systems", Systems Development Corporation, Santa Monica, California, May 1969.

6. Karush, A.D., "Benchmarking Analysis of Time-Sharing Systems", Systems Development Corporation, Santa Monica, California, April 1969.

7. Lasser, D.J., "Productivity of Multiprogrammed Computers - Progress in Developing an Analytic Prediction Method", Communications of the ACM, v.12, No. 12, December 1969, p.679.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Assistant Professor G.H. Syms, Code 53 Zz Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	Assistant Professor G.E. Heidorn, Code 55 Hd Department of Operations Analysis Naval Postgraduate School Monterey,California 93940	1
5.	LT William R. Haines, USN 38657 Rhons Wood Court Northvill, Michigan 48167	1
6.	LT James H. Porterfield, USN 204 Algeria Road Fort Ord, California 93941	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3 REPORT TITLE

An Empirical Comparative Investigation of the CP/67 and TSS/360 Time-Sharing Systems.

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; June 1971

5. AUTHOR(S) (First name, middle initial, last name)

William Robert Haines and James Harold Porterfield, Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1971	64	7

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

A set of terminal test programs called a benchmark has been derived for the purpose of comparing the two time-sharing computer systems, CP/67 and TSS/360, in order to determine which one can best meet the present operational requirements at the Naval Postgraduate School. Some of the problems encountered in attempting to design the benchmark are discussed, along with the problems of trying to make a valid comparison of the two systems. The results obtained from a series of tests conducted over a period of six months are compiled, analyzed and presented in a manner which shows that CP/67 is significantly superior in most respects to TSS. In addition to the comparison of the two systems, information is presented which spotlights many of the problems which degrade CP/67, and recommendations for the alleviation of those problems are made.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31406

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Time-Sharing Evaluation						
Benchmarking						
CP/67						
TSS/360						

Thesis 128398
H124 Haines
c.1 An empirical compar-
ative investigation
of the CP/67 and
TSS/360 time-sharing
systems.
24 NOV 71 19755
29 DEC 86 31377

Thesis 128398
H124 Haines
c.1 An empirical compar-
ative investigation
of the CP/67 and
TSS/360 time-sharing
systems.

thesH124
An empirical comparative investigation o
3 2768 002 13671 5
DUDLEY KNOX LIBRARY